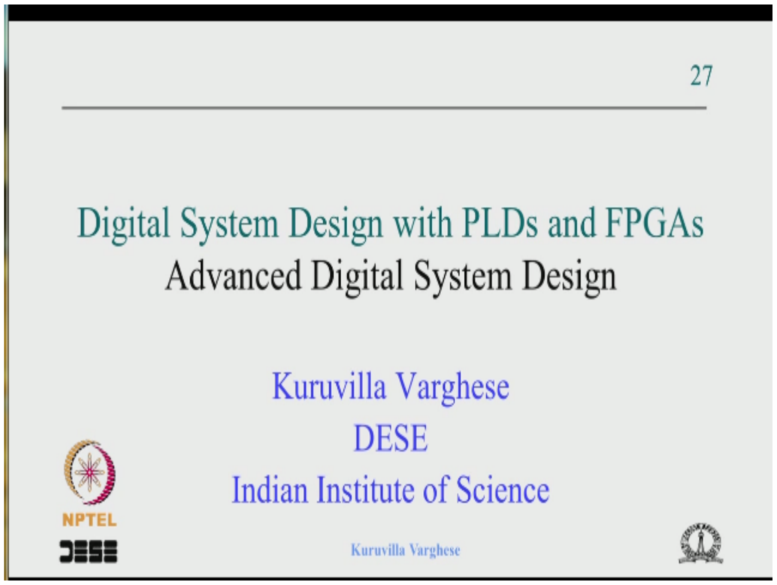
Digital Systems Design with PLDs and FPGAs

Kuruvilla Varghese
Department of Electronic Systems Engineering
Indian Institute of Science - Bangalore

Lecture-29
Synchronization 1

So, welcome to this lecture on advance digital system design**.**

(Refer Slide Time: 00:26)



27

Digital System Design with PLDs and FPGAs
Advanced Digital System Design

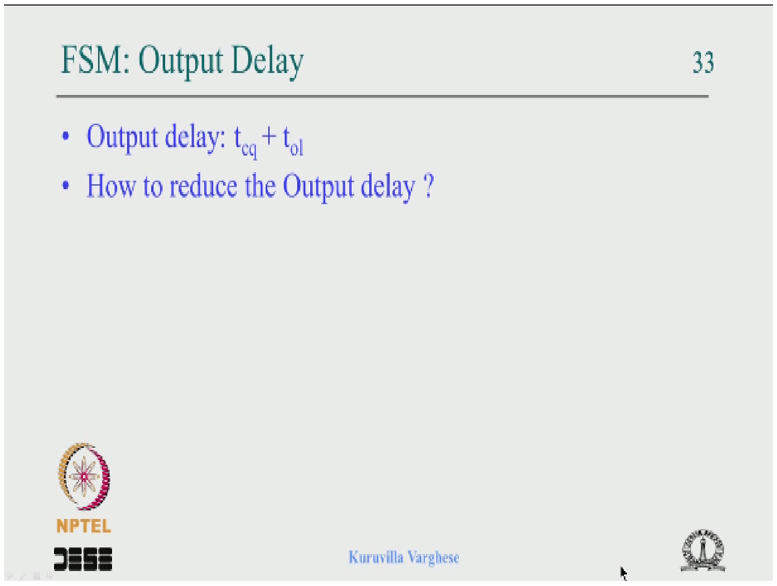
Kuruvilla Varghese
DESE
Indian Institute of Science

NPTEL

Kuruvilla Varghese

In the course digital system design with PLDs and FPGAs in the last lecture we have looked at how to reduce output delay in finite state machine. They were 2 techniques; basically to decode the output form the next state instead of the present state the second technique was to encode the output in the present state okay. So, quickly before to getting into today's portion. We will run through the slides quickly.

(Refer Slide Time: 01:08)



FSM: Output Delay

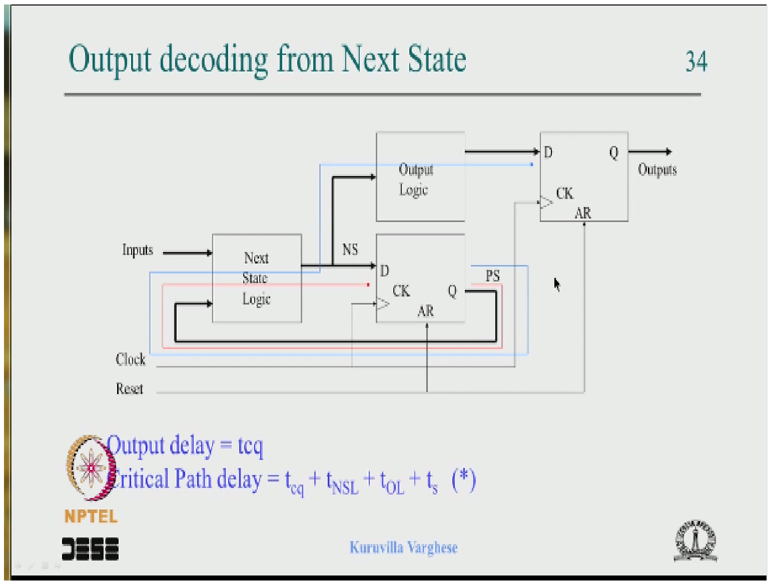
33

- Output delay: $t_{cq} + t_{ol}$
- How to reduce the Output delay ?

Kuruvilla Varghese

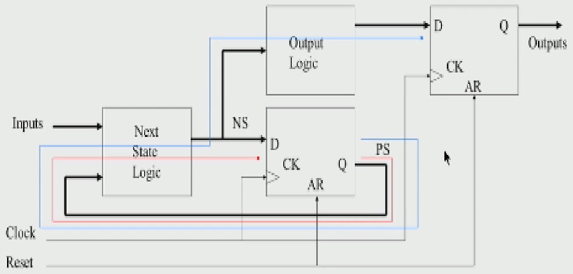
So, let us look at this so, basically our aim was to reduce the output delay, and these are the two techniques, we said output decoding from next state.

(Refer Slide Time: 01:14)



Output decoding from Next State

34



Output delay = tcq

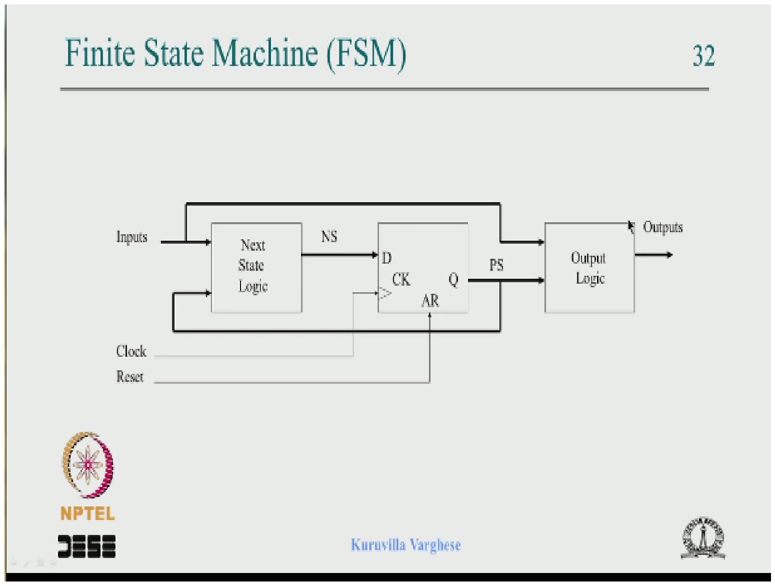
Critical Path delay = $t_{cq} + t_{NSL} + t_{OL} + t_s$ (*)

Kuruvilla Varghese

So, in normal course of time we decode the present state to generate the output, our idea is that hopefully there is a you know if you put the output logic here. And followed with the set of flip-flops so, that when the clock comes you know you get the output along with the present state. So, the only output delay is tcq that is the idea so, that is how you know we have put it from the next state logic here our output logic and the output flip-flops.

Now there are two path, 2 register to register to two different kind of paths. There are many paths may be if there are 3 flip-flops you know 3 into 3 9 if there are two outputs 3 into 2 6 paths in this side. But here at least by the number of terms this particular path has more terms tcq, tnext state logic, toutput logic and the set up time okay, earlier one advantage is that this is coming. You know kind of one after the others. So, it could be optimise earlier if you look at the earlier diagram.

(Refer Slide Time: 02:32)



Finite State Machine (FSM)

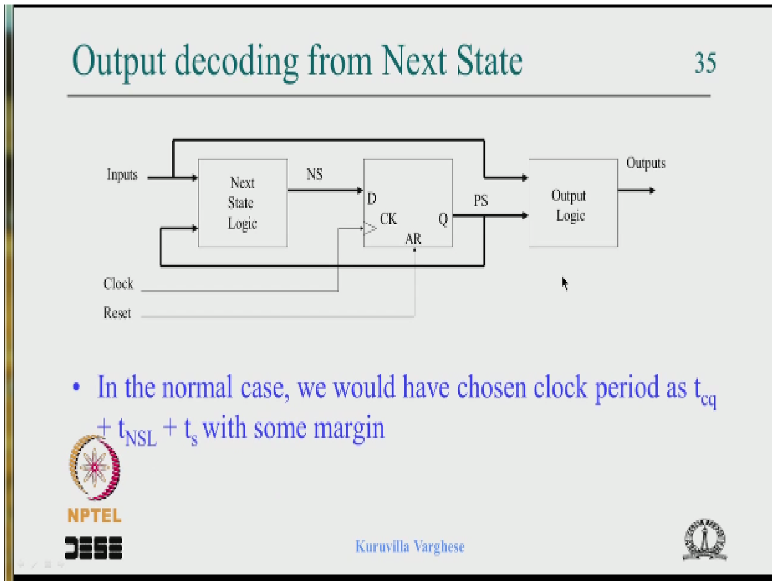
32

NPTEL

Kuruvilla Varghese

This output logic was after the register boundary. So, there is no way it can be combine with this. So, there is no kind of optimisation possible there. But it may happen that this to put together can become as equal to the earlier one, then we have an advantage, because everything is same the clock cycle is same, clock cycle is not affected and the output delay is reduced.

(Refer Slide Time: 03:05)



Output decoding from Next State

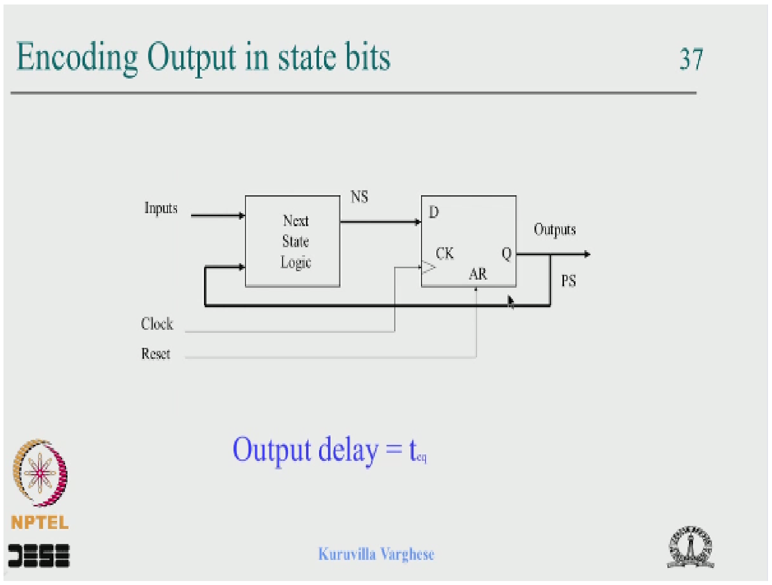
35

- In the normal case, we would have chosen clock period as $t_{cq} + t_{NSL} + t_s$ with some margin

Kuruvilla Varghese

So, that is the advantage of this scheme.

(Refer Slide Time: 03:18)



Encoding Output in state bits

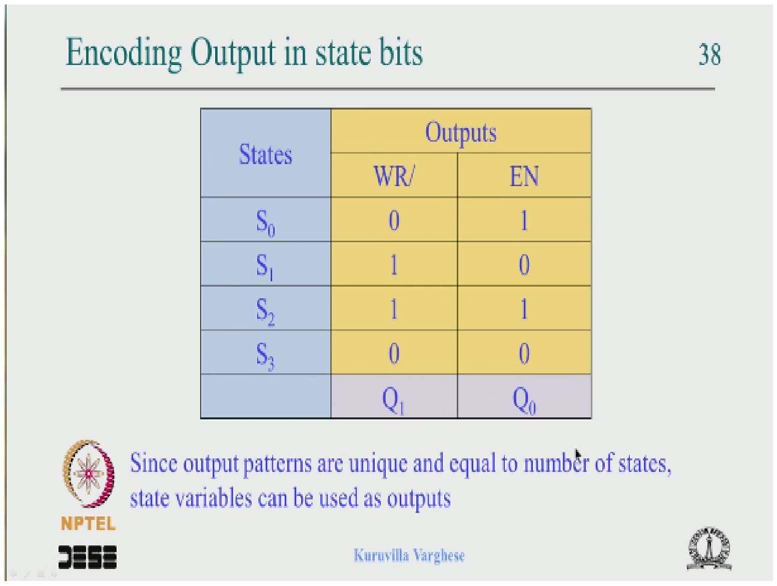
37

Output delay = t_{cq}

Kuruvilla Varghese

And the other scheme is to use the present state as output definitely there are lot of points to kind of ponder because how many number of outputs are there how many number of state flip-flops are there what is there one could be greater than the other one and things like that.

(Refer Slide Time: 03:36)


Encoding Output in state bits

38

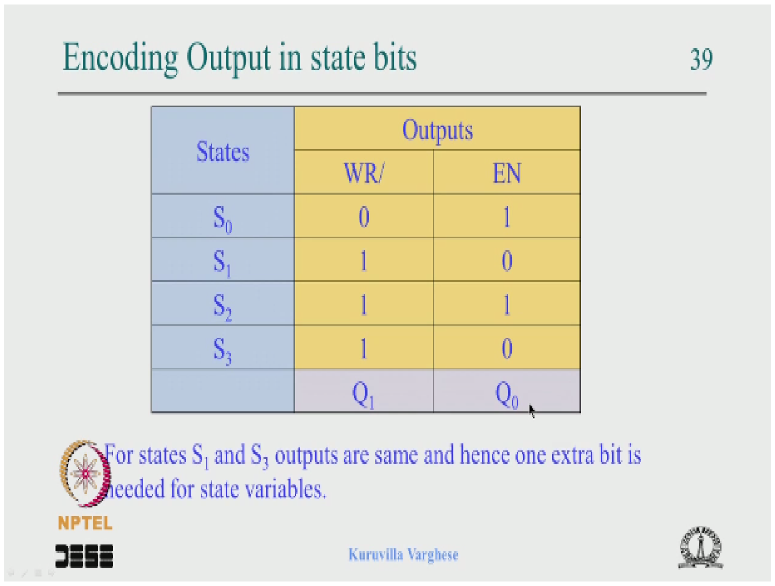
States	Outputs	
	WR/	EN
S_0	0	1
S_1	1	0
S_2	1	1
S_3	0	0
	Q_1	Q_0

Since output patterns are unique and equal to number of states, state variables can be used as outputs

Kuruvilla Varghese

But at least we have seen that when the number of outputs are like equal to the number of flip-flops and if the pattern is unique for each state straight away we can use one of the flip-flops as **as** for one output when other one for the second output, there is no harm.

(Refer Slide Time: 03:58)


Encoding Output in state bits

39

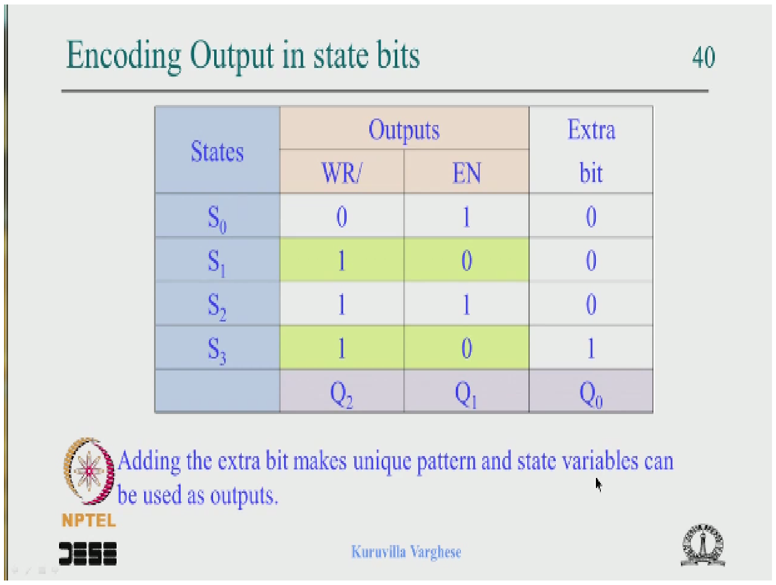
States	Outputs	
	WR/	EN
S_0	0	1
S_1	1	0
S_2	1	1
S_3	1	0
	Q_1	Q_0

For states S_1 and S_3 outputs are same and hence one extra bit is needed for state variables.

Kuruvilla Varghese

But if there is any kind of reputation like in this case S1 and S3 are the same output pattern okay. So, in that case we will be force to add an extra bit.

(Refer Slide Time: 04:08)



Encoding Output in state bits

40

States	Outputs		Extra
	WR/	EN	bit
S_0	0	1	0
S_1	1	0	0
S_2	1	1	0
S_3	1	0	1
	Q_2	Q_1	Q_0

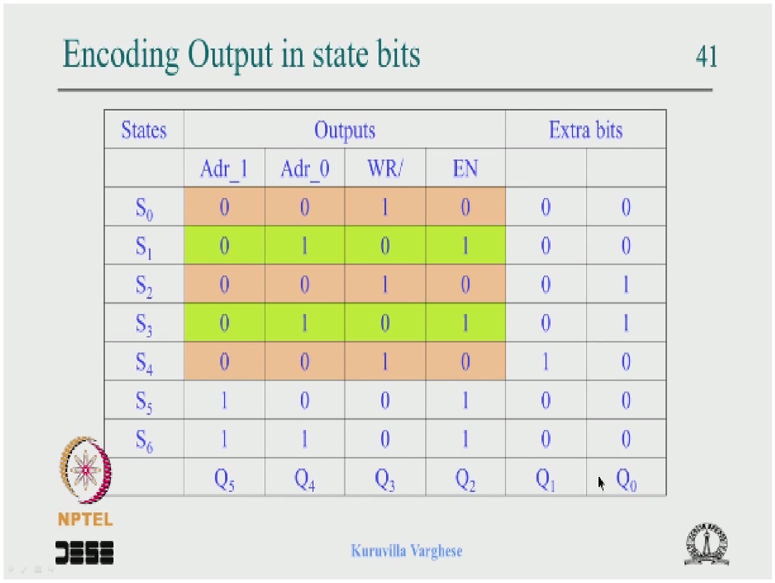
Adding the extra bit makes unique pattern and state variables can be used as outputs.

NPTEL

Kuruvilla Varghese

That means the state present the state flip-flops will be 3 flip-flops though we have only for states this is to decode the output from the state, because there are reputation as for as the output pattern is concern in this case. So, we add extra bit to kind of de-market them. And but still we end up having at the cost of 1 flip-flop our output delay is reduced.

(Refer Slide Time: 04:43)



Encoding Output in state bits

41

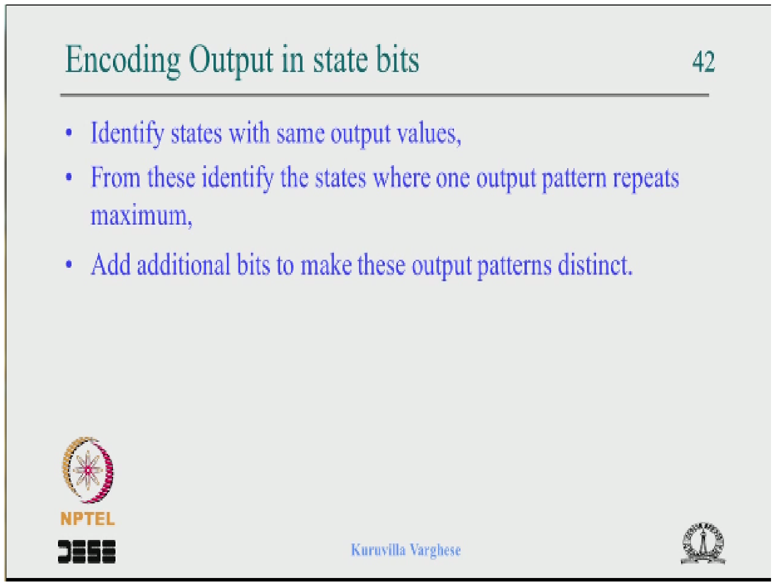
States	Outputs				Extra bits	
	Adr_1	Adr_0	WR/	EN		
S_0	0	0	1	0	0	0
S_1	0	1	0	1	0	0
S_2	0	0	1	0	0	1
S_3	0	1	0	1	0	1
S_4	0	0	1	0	1	0
S_5	1	0	0	1	0	0
S_6	1	1	0	1	0	0
	Q_5	Q_4	Q_3	Q_2	Q_1	Q_0

NPTEL

Kuruvilla Varghese

And we have seen another example much more complex example where one output pattern is repeated twice and one 3 times then we have force to add 2 extra bits to make a difference like 00, 01, 10 and as I said there are 5 state flip-flops or the present state flip-flop that you know account for like a you know 2 rise to 5, 32 pattern. But we have only 4 output there are only 2 rise to 4 possibilities. But because the output is repeating in state we are force to add this 2 extra bits you know.

(Refer Slide Time: 05:29)



Encoding Output in state bits

42

- Identify states with same output values,
- From these identify the states where one output pattern repeats maximum,
- Add additional bits to make these output patterns distinct.

NPTEL

Kuruvilla Varghese

So, the rule is that you identify the state with same output values from these identify the state where 1 output pattern the repeat maximum. And additional bit to make the output pattern to sink you do not have worry about whether something is greater than or less than. If you try this algorithm this will work out and it is a good thing to try, you know it is a really good thing to try the tools may not do it.

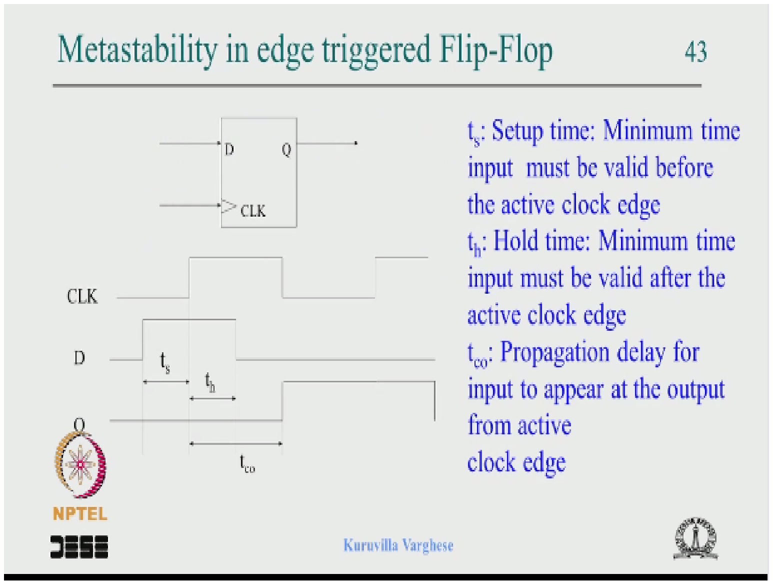
Because you know that you have already seen that the various attributes the tools normally allow the sequential kind of encoding the grey coding or the 101 and things like that, 101 as a possibility of this happening, or because output can be delay can be reduced. We will see that later, when we look at the FPGA so, that is what we have seen the last lecture. Today we are going to look at a serious issue called synchronisation okay.

And that is related to an issue in a flip-flop called metastability, basically it is not it is a issue inherent in a cross couple latch okay. Since we built flip-flop out of cross couple latch this same problem comes in a flip-flop. And we built data path and the finite state machine using flip-flops so, we are again have the same problem. So, it all start with latch.

But we do not have you know time to look at from the transistor level, or at the inverter level the bound to here okay, actually normally if you learn that way, you can get a good grip on it. But we will abstract it as for as the flip-flop is concern, what is the rule to follow and I will tell a just of it why it is happening. And how to handle it that should equip you to handle most of the situation you come across.

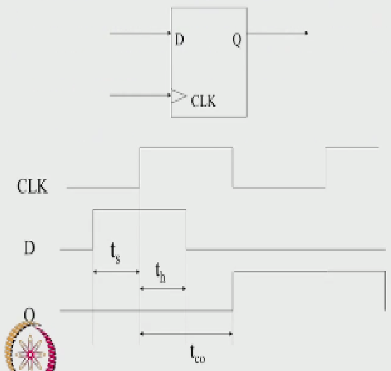
We are not again discussing all possible advance technique to do the synchronisation, which I hope you to follow it up with the basic and this course, we are already you know it contains quite a lot of things now to add this is may be little too much of a scope so, let us look at the problem of meta stable team the flip-flop.

(Refer Slide Time: 08:17)



Metastability in edge triggered Flip-Flop

43



t_s: Setup time: Minimum time input must be valid before the active clock edge

t_h: Hold time: Minimum time input must be valid after the active clock edge

t_{co}: Propagation delay for input to appear at the output from active clock edge

Kuruvilla Varghese

So, we have already learn that in an edge to that flip-flop so, you have a clock d and q normally a behavioural kind of description is that, when the clock edge comes whatever is at the d is propagated to q with the delay. But we know that for that to happen properly okay, the data here as to arrive some time before the active clock edge there in the case suppose the edge, and that time is called set up time or Ts not only that the data after the clock edge.

It has to remain there for sometime that time is called hold time. So, unless the data the input need this set up and hold time, the q will not get this copy of this d, copy or the value of d. The d will not be transfer to q properly okay so, that is what is I mean we have many a times we have learned that for successful transfer of operation of the edge to the flip-flop.

You have to meet just a set up, and hold time at the input many a times, it is discuss what happens if it is not met okay. So, that is what we are going to do today now the game is that so, this is the essential behaviour of the flip-flop. The data should arrive set up time before the clock edge, data should remain kind of hold time after the clock edge, active clock edge. And if so, happens from the active clock edge with some delay this output appears okay.

Now that means that though we say in a functional way many a times when you learn at the beginning you draw kind of the waveform without delay there whatever you know the edge, whatever is the value at the edge. It is transfer to the output, you know that is how it is done at a behavioural level, but there is no magic know there is nothing can work like a magic just an edge comes the output cannot instantaneously goes there.

And you also would have studied a master slave flip-flop, which is working as x to get flip-flop, because most of you would have learn master and slave configuration. And that is a essential logic circuit for h2 get flip-flop and most textbooks give 2 cross couple latch one after the other. So, during the negative clock period the master is in the transparent mode when it goes positive.

The master is cut off and the slave, whatever is capture in a master is transferred to the slave and you get the output okay. So, there is no magic, there is no like edge to trigger magic you know as soon as the edge comes you know as if something happens during the edge you know. Nothing happens during the edge in digital you know that either it is 1 or 0. So, something happen to the master latch during the clock period is low.

Something happen to and that is cut off, something happen to the slave latch during when it is high, whatever is captured in the master is transferred to the slave okay. So, that is why this setup and hold time come and the as I said I am not going have an analysis due to lack of time. But then you can definitely go back, and look at the architecture of the master slave not only, what you have learned any master slave.

There are different you know master slave architecture though something are not call the master slave you can take the like if you remember the 7, 4 TTL series you can take 7, 4 D7, 4 flip-flop and look at the internal diagram you will find the there is a kind of master slave architecture which is not same as what you have learned in in the textbook.

And the number of gates used will be less and the propagation delay is also will be less in those configuration. But the analysis is same, so what I am saying is that the game in a gist you show that you know I do not leave some curiosity kind of flip-flop (()) (13:07) like, if I say accept as it is and you do not know, where it is coming from suddenly a magic. So, I **j**ust want to tell that when the clock period is low the master latch will be in the transparent mode okay.

So, master latch is capturing the input. But the latest when it can capture is sometime before this is getting cut off. Because when this edge come, and when it turns positive master input is cut off okay. So, **so** but the master latch to kind of set there is a cross couple latch delay Okay. So, that delay before this clock edge is a setup time, and we have learned that there will be, because the master is enable during the negative the slave is negative period.

The slave is enable during the positive clock period. So, there will be invertors in the clock path. So, there will be like the normally you will have the master latch delay – inverted delay will be the setup time. Because that is the latest a master latch can set before the clock edge happens okay. Now the clock edge comes, now because of the invertors in the clock path it take some time to cut it off, input to cut it off.

So, during that time nothing should happen to the D, then the master latch will upset. So, upset in the sense value will be upset**.** So, there is a hold time which is equal to the, you can say the inverted delay in the clock path. So, that is a hold time. Now when this clock period is enable. The slave latch will take some time to set that is the propagation delay okay.

So, that is how this comes you know as I said during the negative clock edge clock period. The master is in transparent mode so, before it is cut off what is the latest the master can latch, that is this setup time. That is a cross couple latch delay–the invertors, any invertors in the path. And when it is cut off, when it goes to positive that inverted delay has to be kind of met.

Because then only it will be a really cut off, that is a hold time, and then when it is you know when it is the slave is enable. The master output is latched on to the slave. So, that latch delay + any inverted delay in the clock path. If you analyse it you will get, that is the propagation delay now. So, that is how this come that means once again to for the D to go to Q, you have meet setup and hold time that is all.

But what happens if this is not met okay, that means the master is getting cut off and you are still changing the input okay. So, what happens then what happens is that again that need little bit of analysis which I cannot kind of which we have to go back to the cross couple latch. So, since as I said the lack of time we will just kind of abstract it what happens is that, if the setup or hold time is not met.

Many things can happen one thing the best can happen is that like suppose the output was 0 here, and you are trying to drive a 1 upon the clock edge. That means we are expecting this 1 will come here okay. But if the setup or hold time is not met. Then we are not sure the, whatever the 0 at the output may remain there 0 or the **y** it may become 1. So, we cannot say maybe, it will change state or it may not change the state okay.

It is still an acceptable behaviour, there is no harm. Because earlier it was 0, we are expecting it to go to 1. Since it is the data is coming in the changing in this window, it is not you know setup it is not the data is not setup properly. So, data is not transfers still it is okay. Because it is 0, this is 1 and we are hoping that next clock edge come. So, if the data remains, and normally the data does not remain just like that data will remain for 1 or 2 clock period.

And the data is transfer properly. So, a miss trigger is a very good thing you know the best thing you can hope for. If the setup and hold time is violated, the best one can hope is that let it miss trigger. That means instead of going to 1 let it be 0, there is no harm. Because earlier it was 0 and the next clock edge it becomes 1 okay. Now another effect of this metastability in flip-flop. This behaviour is called metastability **.**

Another effect of that is that if the data is change in within this window close to the clock edge it will take the output will either become 0 or become 1. But it will take long time to resolve like that say output was 0, it was going to 1, but it would not transit fast it will take some time and settle it to 1 okay. So, that is another issue which is already bad, because in a register to register path. We say t clock is greater than tcq+tcomb+tsetup.

And we have fixed the maximum tcq. Now if it is takes long time to resolve, that will violate the setup time at the destination flip-flop, and that can cause meta stability there okay. So, it can propagate this all game can propagate from 1 flip-flop to the other it is very bad. The worst case you know, so 2 things either it miss trigger it takes long time to miss trigger like even for output to resolve it takes long time.

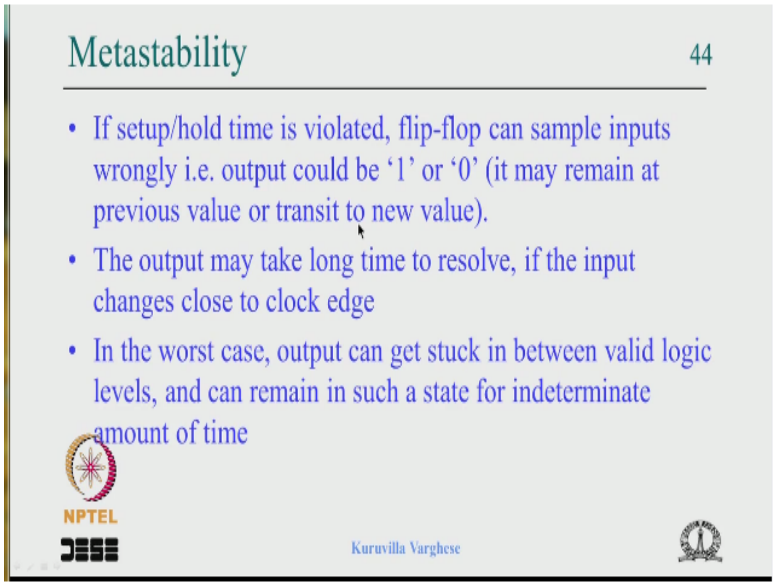
Third effect is that it can get stuck in between the logic levels. Suppose you know that in a 5 volt the say the lowest high value is a 3 volt. And the highest low value is 2 volt we will expect always the logic 1 to be greater than 3 volt, and logic 0 to be less than 2 volt. But what happens is that in the case of metastability, it may happen that such a flip-flop can have a output which is 2.5 volt okay.

Now that is very dangerous if that 2.5 volt goes to a logic circuit, a combination circuit what can happen is that it can drive the transistors inside to active region. So, instead of kind of a switch latch is an amplifier any noise it will drive to saturation. So, it will stop switching okay. So, there could be lot of errors, if it is not in the kind of design it will not taken care. This metastability can cause lot of errors okay.

The miss trigger is very safe, because as I said it may go to 0 or 1, still it is okay. But if it gets stuck in between and that whatever is being driven can also get into metastability. So, it is a disasters thing and many a time this issue is not taught in the basic course. But it is a very serious thing, you know I am most of the time the design practices a noise use does not consider the metastability.

It is a really bad thing to do. But most of the time because of the reliability these devices are very fast and the probability of that thing happening is less, we are same, but this should be kept in mind. So, that is the basic issue, what is called metastability okay. In an x to get flip-flop if the setup for hold time is not met, we are not sure of output. It can get a it can take longer to resolve, it can gets stuck in between which is the worst kind of thing.

(Refer Slide Time: 21:35)



Metastability

44

- If setup/hold time is violated, flip-flop can sample inputs wrongly i.e. output could be '1' or '0' (it may remain at previous value or transit to new value).
- The output may take long time to resolve, if the input changes close to clock edge
- In the worst case, output can get stuck in between valid logic levels, and can remain in such a state for indeterminate amount of time

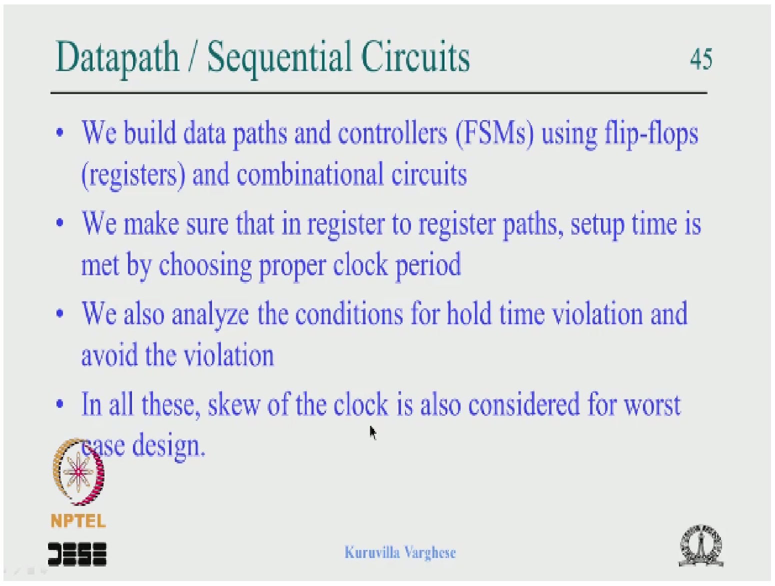
NPTEL

Kuruvilla Varghese

So, that is what I have summarise here, if setup hold time is violated flip-flop can sample input wrongly, output could be 1 or 0, that means in the case, it was 0 before it may remain at previous value or transit to a new value. Output may take a long time resolve, if the input changes close to clock edge. The worst case output can get stuck in a non logical value. And it can remain there in such a state for indeterminate amount of time.

We do not how long it takes, you know maybe it will come out of it after 1 second, 1 hour **,** 2 hour, 1 day we have not sure. We cannot say answer probabilistic thing you know we cannot definitely say that this will happen or if it is happens it comes out with certainty okay, it is a **a** probability. So, that is metastability**.**

(Refer Slide Time: 22:33)



Datapath / Sequential Circuits

45

- We build data paths and controllers (FSMs) using flip-flops (registers) and combinational circuits
- We make sure that in register to register paths, setup time is met by choosing proper clock period
- We also analyze the conditions for hold time violation and avoid the violation
- In all these, skew of the clock is also considered for worst case design.

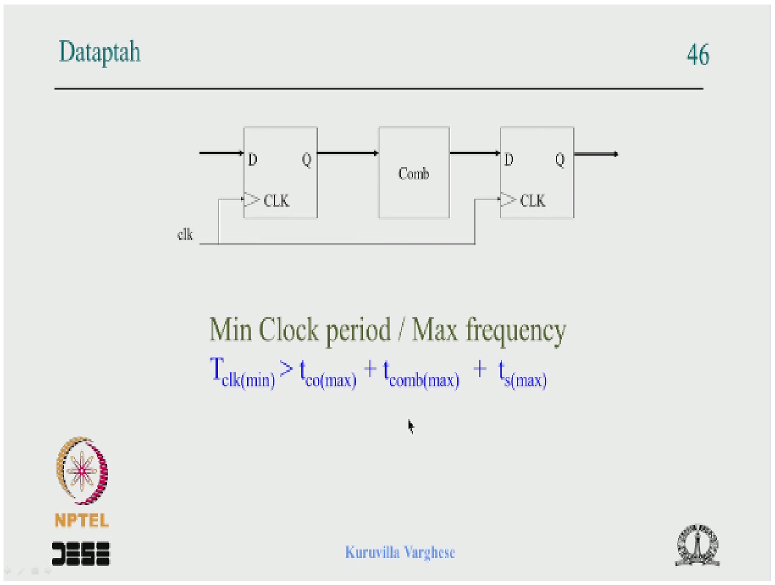
Kuruvilla Varghese

But let us come back to the data path and sequential circuit. As I said we use flip-flops, and combination circuit okay. So, all the data path are built of the flip-flops, registers and combination circuit similarly the FSM also use the registers, the state registers and the combination circuit next state logic, output logic and so on. So, now this as potential chance of like probably there is a chance of flip-flop getting into metastability.

But when we design we analyse all the register to register path, and we choose like it at the destination register, you always try to meet the set up time by choosing the proper clock period like we will say t clock is greater than that path delay. That is taken care it mean the analysis while simulating all that. We make sure that the clock period is not exceeded, similarly at the destination register. We also make sure that the hold time is not violated but it is violated, we add extra combinational delay and sorted out.

You know so, we take care of these both condition and if there is a q of the clock, which we have not I think formally learned in this course which we will do when you go to the FPGA. I will clock that part in the case of the FPGA, because there is a reason to do that. We will see that, when we learn I mean how to take recue of the clock in the analysis. We will see that okay, but a proper designer takes care of the clock period, take care of rescue, take care of the hold time violation again with this the rescue and all that.

(Refer Slide Time: 24:31)



Dataptah

46

Min Clock period / Max frequency

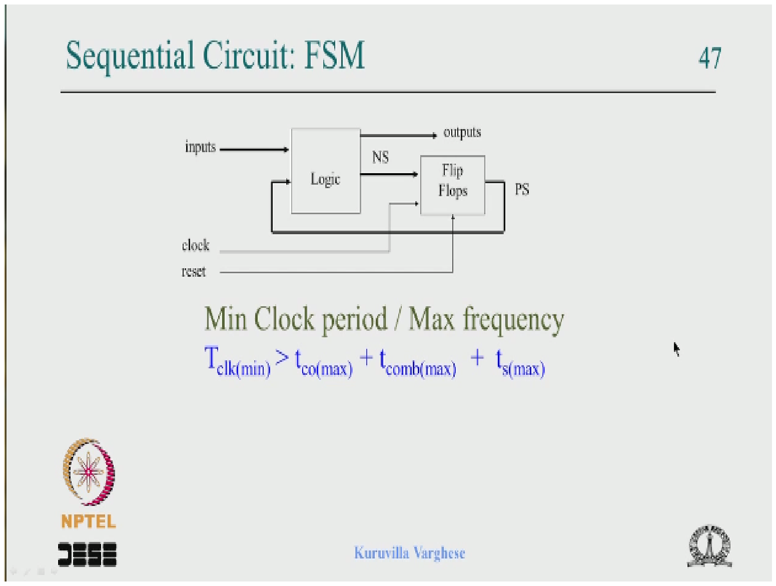
$T_{clk(min)} > t_{co(max)} + t_{comb(max)} + t_{s(max)}$

NPTEL

Kuruvilla Varghese

So, everything is done properly so, you have a date path, register, combination circuit, register we say okay the clock period should accommodate the tcq, tcomb and the set up done. The clock period is 1000, and if there is rescue that is taken care, similarly to avoid the hold time violation here. We have to make sure that the minimum delay of this flip-flop + the minimum combination delay should be greater than the minimum hold time. So, that is also taken care.

(Refer Slide Time: 25:01)



Sequential Circuit: FSM

47

Min Clock period / Max frequency

$T_{clk(min)} > t_{co(max)} + t_{comb(max)} + t_{s(max)}$

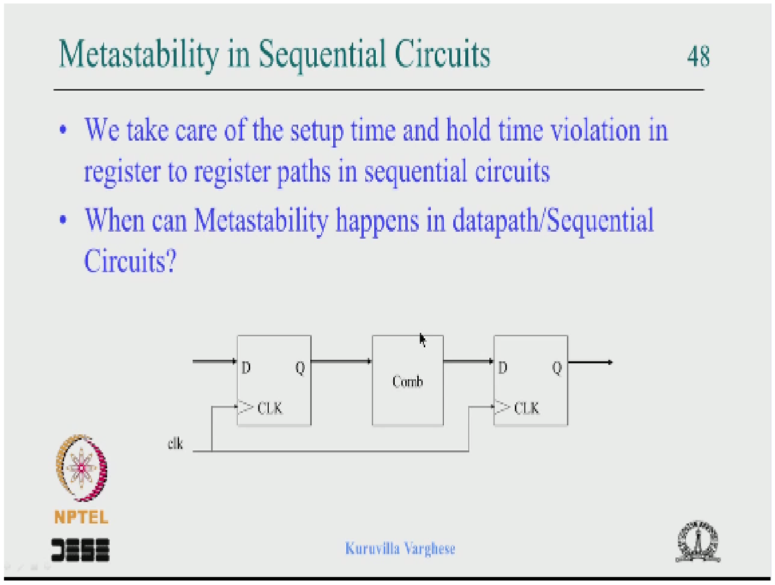
NPTEL

Kuruvilla Varghese

Similarly with the FSM, you have the logic here decoding the next state and output from the present state, and we again make sure that the clock period is met Tclock is tco+tcomb+tlogic+tsetup time and the hold time again tcq, tlogic, tcomb minimum it should be greater than the thold minimum. This all taken care so, what is why are be worried about metastability okay. So, the question is that we take care of in a data path in a FSM.

We take care of the destination register, set up time and the hold time okay. So, why like that is **that is** a essential part of the design the timing design, then you do we worry about the meta stability in flip-flop. Because it is only violated then the meta stability happens.

(Refer Slide Time: 25:59)



Metastability in Sequential Circuits

48

- We take care of the setup time and hold time violation in register to register paths in sequential circuits
- When can Metastability happens in datapath/Sequential Circuits?

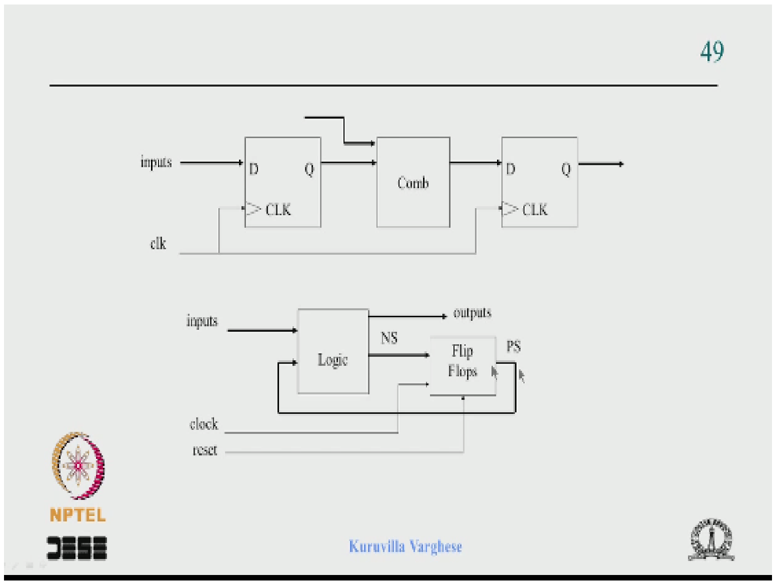
D
Q
CLK
Comb
D
Q
CLK
clk

NPTEL

Kuruvilla Varghese

So, that is the question we take care of the set up time and hold time violation in register to register path in sequential circuit at data path. And when can metastability happens, in the data path okay.

(Refer Slide Time: 26:12)



So, that is the question so, look at this picture then you might get it like or look at this picture where things can go wrong like, we have analyse this path. You know from something clock comes, the output change it propagates through here and it come here nothing can happen. Because we have taken care the clock period is greater than this path delay so, where things can go wrong that is so, we will let us put this picture of the data path of the sequential circuit.

Now look at where things can go wrong, and am I also showing some input here directly coming into this combination circuit. Now you know that we have some input to this register, we have some input to the combination circuit, and we are not sure where it is, this is coming from similarly the state machine as inputs external inputs maybe some inputs are coming from the data path. Some inputs are coming from some other place.

So, now the cracks of the problem is this these inputs, we are not sure where it is coming from it is okay if all the inputs are coming from a register clock by the same clock. But assume your data path as 2 clocks and this is the clock, and there is another clock 2, and there is a register which is clock by the clock 2 that output is coming here. Then that as know face relation with this clock, and this data you know propagating to the combination circuit may not be the set up time here okay.

So, there could be a problem of meta stability here okay, similarly these inputs we do not know where it is coming from maybe it is not from the same clock or it is coming from a process which is not related to this clock say example is that something is moving, and there is a limit switch. It touches a switch and that switch output is coming saying that, the limit is reach here. So, that probably that movement is driven by a motor which as got nothing do this clock.

So, that process as know relation to the face of the clock here so, it can violate the set up time at this input. So, whenever there are inputs which is not synchronous to the clock, your clocking the registers. They can cause metastability in the respective register. So, this input can cause metastability here. This can cause metastability in this particular input flip-flop okay, now look at the state machine once again this.

The state flip-flop to the state flip-flop path absolutely no problem. We have taken care of the clock period the hold time this q and so on. But you see there are inputs from maybe from the data path which is clock by the same clock. But maybe one of the input is coming from elsewhere which has no face relation to this clock. Then that can cause meta stability in this flip-flop, because it propagates through it does not matter.

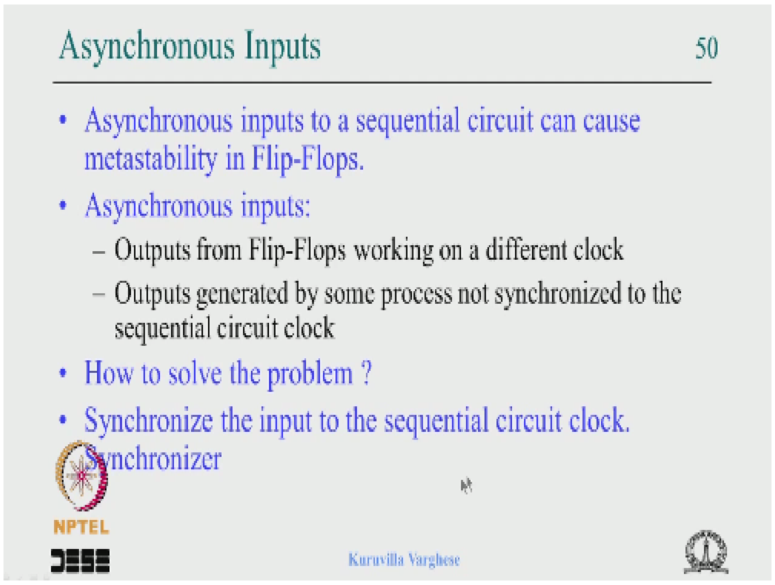
Because this is coming randomly, this logic want do any good you know like if it is quite random with the clock. This logic delay I mean does not it makes not difference okay. Because the hold pattern is shifted by a constant delay as for as the kind of the probabilities are concerned it does not accept. The probability of a setup hold violation. Because a delay will not do any good.

So, these inputs which are not synchronous to the clock or the culprits of meta stability in a sequential circuit. So, and we called that as asynchronous inputs okay. So, asynchronous inputs in a data path like this or like this or in a sequential machine, sequential finite state machine. The asynchronous input can cause metastability in the flip-flop okay. Now the name like the solution is in the name itself.

The metastability happens, because the input is not synchronous, input is asynchronous so, if somebody ask you what is the solution like you know even if you are you can say yeah asynchronous input is causing the meta stability so, synchronise it okay like if it is synchronous it does not happen you can boldly say synchronise.

Even if do not know how to synchronise so, these are this is one of that type of question to answer it is very easy. But you may not know how to do it okay but maybe in interview you can kind of impress people maybe that question can give you a good job. You know is a like why metastability happens in sequential circuit is asynchronous input what is the solution is synchronous synchronise it. You know then you get the gob you get a good salary.

(Refer Slide Time: 31:45)



Asynchronous Inputs 50

- Asynchronous inputs to a sequential circuit can cause metastability in Flip-Flops.
- Asynchronous inputs:
  - Outputs from Flip-Flops working on a different clock
  - Outputs generated by some process not synchronized to the sequential circuit clock
- How to solve the problem ?
- Synchronize the input to the sequential circuit clock. Synchronizer

NPTEL

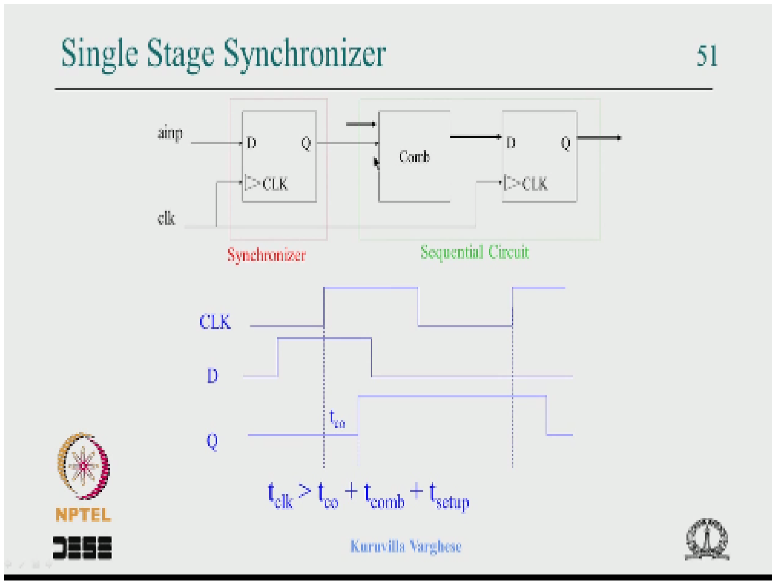
Kuruvilla Varghese

So, we will see how to kind of you know synchronise it so, when I want to bring clarity to that asynchronous input to a sequential circuit can cause meta stability in flip-flop. When we say asynchronous input it is a output from flip-flop working on a different clock reaching the destination. It is output generated by some process not synchronise to the sequential circuit clock.

Yeah as I said maybe something is moving which is touching a switch that movement has got no relation with the clock which we are clocking the input register so, that those are asynchronous input how to solve the problem, as I said synchronise the input to the sequential circuit clock okay. That means that you have to the there are edges which is changing randomly input is changing any time.

We have to make it change with respect to the clock of the our system clock like we have a sequential circuit are a data path. Now we have to make sure that this input which was kind of no face relation to the clock. We are clocking the data path sequential circuit, we have to make sure that this input change in face okay with the constant face relation to this clock okay.

(Refer Slide Time: 33:10)



Single Stage Synchronizer

$t_{clk} > t_{co} + t_{comb} + t_{setup}$

So, that is simple, because suppose assume this was the kind of scenario. This input was coming from outside going through the combination circuit to the some flip-flop in a sequential circuit. Now this was asynchronous okay, now how to synchronise it what we do is that, we put a d flip-flop put that asynchronous input connect that to the d of the flip-flop. And the q you take it and give it to this particular combination circuit.

So, this is the target sequential circuit or data path. This is the synchroniser, we call it as a single state synchroniser. Because there is only one flip-flop okay, now if you look at the clock wave form a clock comes and whatever is the analog input here. If it like if it is like this then it appears here with the delay of tco every time. You know anything even if it changes before upon the clock edge only changes the state.

And you know that now it is becomes synchronous, because every time after the clock, it appears neatly with the delay. Now we are sure that, when the next clock edge comes here it is setting up properly. Because much before that the data is set up here, now you have the time for it propagate through the combination circuit and need the set up time okay. So, that is how it is synchronise it is same now, now like you have it tcq+tcomb+tset up that should be less than the clock period okay.

So, we have kind of synchronise, we made sure that irrespective of the input change here. The output is synchronise to the clock edge so, just a flip-flop that is all what we need to synchronise okay. But fine I mean that sound great you know the really great once again a clever student will ask a question is the problem solved. Now we said that this is changing with no relation to the clock and that comes here.

And upset the destination flip-flop causing meta stability here. But now we have the same issue here, we have put a flip-flop, and said fine now the data appears with the delay. And it is the set up time is met here, but what about this flip-flop okay. Now there the data at the input of this flip-flop has no relation to the clock here. So, now this can get into metastability so, we are not kind of solve the problem.

And looks like this is the kind of cheating, you know you shifted the problem okay. So, we did not solve the problem, we shifted the problem the question to ask is a game is that a good thing to shift the problem at the outset at least that is a problem with a pictures. You know I have drawn a picture at the kind of a row impression is that, we have not kind of improved upon the situation. But it is not so, because in a like you see there is a thick line here.

And there is a very thin line here okay that should give you a clue in a sequential circuit. There could be 20 flip-flops, and asynchronous input reaching the 20 flip-flops can put all the 20 into meta stability okay. But now we are catching that analog input in a single place okay, now earlier it might propagate through different combinational delay know, we know that when you say a comb here. We are multiple path if there are 20 flip-flops here, there are 20 different paths.

And depending on the delay we have sure that maybe at least few of the flip-flops can get into meta stability. Because the path will a could be different, but now that means this input is sampled at different time by different flip-flop. So, with the random behaviour of this input, we are sure that one or two can get into meta stability say, I mean I am say under the course. Because we have to really see the probabilities, which we have not time to analyse in this class.

But I suppose you get the picture but then the situation is improved, because we are put of single flip-flop and sampling it. It is tightly control there is no path delay, it is directly coming here, and if at all a flip-flop get into meta stability is this flip-flop not all these okay. Now the only hope we have is that instead of all the flip-flops getting into metastability. We have a single flip-flop might get into metastability.

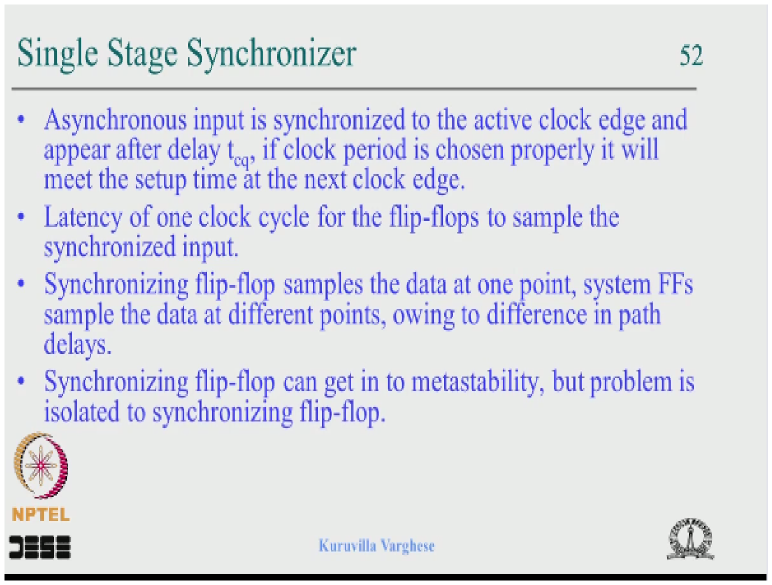
And we are hoping that before the next clock edge, it comes out of metastability okay. That is the bus hope you can have in a single state synchroniser that, it may happen that this all the problem it may happen that, this get into metastability. But if it gets into metastability it comes the output comes out of meta stability with the proper margin.

So, that it can propagate through the combination circuit, and need the set up time for the next clock edge okay. So, we assume that definitely the analog input cannot be a very narrow pulse. It has to be a at least one clock period duration so, that even if it misses 1 it is cot in the next okay. So, naturally and we know that because we have put a flip-flop here it has to be there for at least one clock period. Then only it can propagate through that.

So, there is a late and see the use of synchronise of will add a late and see so, it will be deducted by the sequential circuit only after a clock. And that also necessity that the input pulse width should be more than one clock period wide you know for to be able to properly propagate here. So, this is how the synchronisation is done we have not completely solve the problem but we will look at it.

You know as I said we have isolated the problem to a single flip-flop a single place sampling is done earlier the sampling was done at multiple places at multiple instance of time, because of the variation in the path delay now it is done in one place okay. So, if it is solve, it solve for everything, if it is not solve, it is not solve for all other flip-flops. So, it is better than the earlier case, earlier we have no control like some might get into metastability some may not, but here it is tightly control.

(Refer Slide Time: 40:29)



Single Stage Synchronizer 52

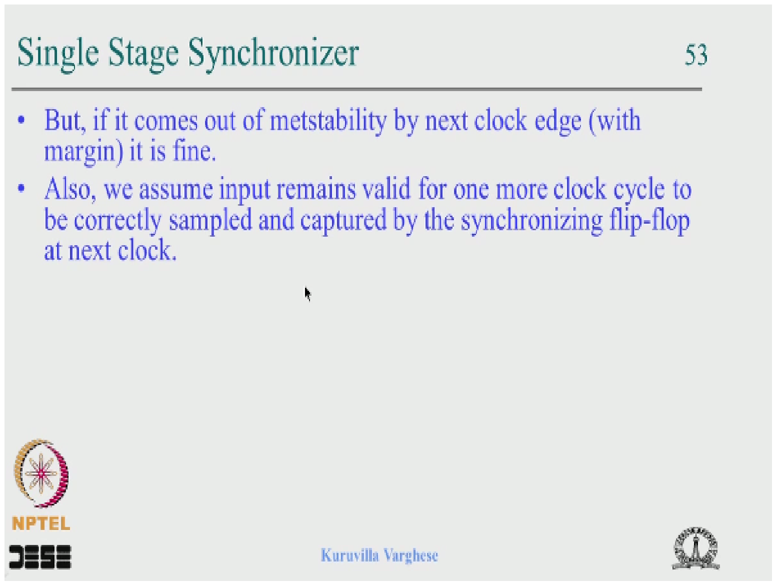
- Asynchronous input is synchronized to the active clock edge and appear after delay t_{cq}, if clock period is chosen properly it will meet the setup time at the next clock edge.
- Latency of one clock cycle for the flip-flops to sample the synchronized input.
- Synchronizing flip-flop samples the data at one point, system FFs sample the data at different points, owing to difference in path delays.
- Synchronizing flip-flop can get in to metastability, but problem is isolated to synchronizing flip-flop.

NPTEL

Kuruvilla Varghese

So, let us come to that, so that is what is listed here. Now the final analysis is that the synchronising flip-flop can get into metastability, but the problem is isolated to synchronising the flip-flop.

(Refer Slide Time: 40:48)



Single Stage Synchronizer 53

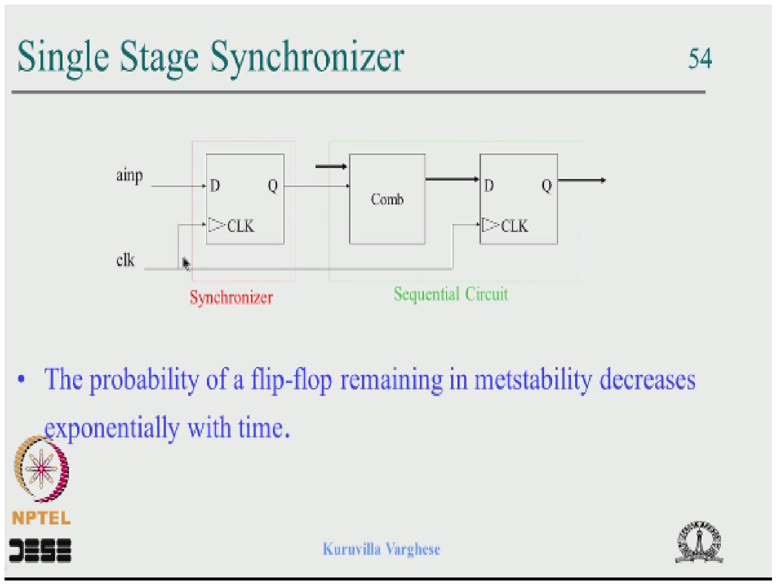
- But, if it comes out of metstability by next clock edge (with margin) it is fine.
- Also, we assume input remains valid for one more clock cycle to be correctly sampled and captured by the synchronizing flip-flop at next clock.

NPTEL

Kuruvilla Varghese

But we are assuming that before the next clock edge is comes out of the meta stability and we have also assume that, the input remains valid for one more clock cycle to be correctly sampled and captured by the synchronising flip-flop and next clock.

(Refer Slide Time: 41:03)


Single Stage Synchronizer

54

- The probability of a flip-flop remaining in metstability decreases exponentially with time.

NPTEL

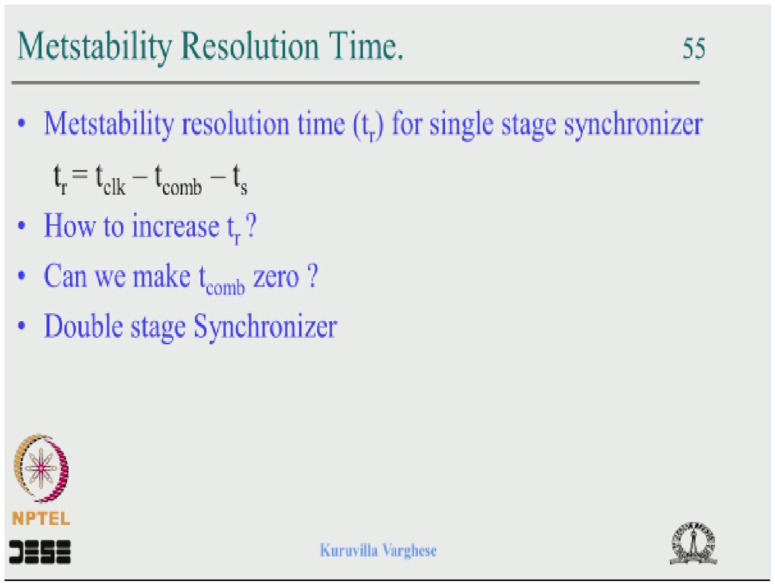
Kuruvilla Varghese

So, now the good news also this is a synthesis state synchroniser, this is synchroniser, this is sequential circuit. The good news is that this the probability of a flip-flop remaining in meta stability decreases exponentially with time okay. Now the mind you read carefully, the probability of a flip-flop remaining in metastability decreases exponentially with time. That means that by if this analogue input sorry.

Asynchronous input violates the setup time or hold time. and if this get into meta stability as the time passes the chances of it coming out of the meta stability it is a chance, it is a probability is higher and higher as you give 1 nanosecond extra the chances are more, you give it more time you have more chances of it coming out of the which probabilistic you not say deterministically 100% of the time it will come out.

But we increase the probability okay. So, the metastability we can only handle probabilistically so, what we do is that we will reduce the probability of you know or we will increase the probability to such an extend that in the life time of the system we are building this happens very rarely. You know that is the basic idea behind it okay so, what is the time available for you know the flip-flop to resolve.

(Refer Slide Time: 42:47)



Metstability Resolution Time.

55

- Metstability resolution time (t_r) for single stage synchronizer

  $t_r = t_{clk} - t_{comb} - t_s$
- How to increase t_r ?
- Can we make t_{comb} zero ?
- Double stage Synchronizer

NPTEL

Kuruvilla Varghese

That time is called meta stability resolution time that is the time meta stability resolution time is the time available for this output to resolve come out of meta stability okay. So, now if you look at it, you have a full clock period available T clock, but you know that once the output get into meta stability. It should comes such that the set up time is met here by the next clock edge.

So, if you have a clock period just be a resolve tclock-tsetup-tcombination delay, then only if it is resolve it will be correctly set up here. So, for a single state synchroniser the meta stability resolution time. That is the time kind of available for the output 2 resolve to a valid state is nothing but that is called tr the meta stability resolution time is tclock-tcomb-tsetup so, that is clear okay. Now say that this has to be ready tsetup +tcomb.

You know which is subtracted from the clock period now that is it, now our will be good if we can increase this okay. So, suppose we like you know this situation is like this we found that you know there is an asynchronous input. We try to synchronise it, we put a flip-flop and we do some analysis. We find the tr and find the probability, that is a analytical expression for doing it. We can model that and get an expression.

Then we find that the probability is not is not very is kind of not very low that is remaining in a meta stability. Then we have to increase the tr so, that to make the probability very low that is a game. So, suppose we calculated this we found the probability, and we find that it is still high, it there is a high probability that will remain. Then we have to kind of increasing.

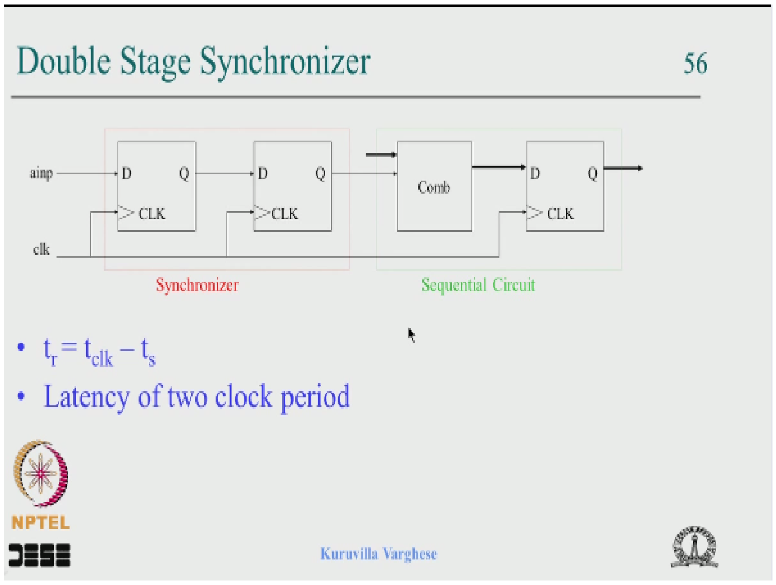
The one way of increasing is that, to reduce the clock period but that is you know that is not on, because we have a when you design a circuit, we have a clock target okay depending on the performance we need like we are doing some computation. And we have a requirement of through put. So, many operation per second, and we know the data path architecture, how many clocks clock cycle it takes for the computation of the result.

We are looking for the, we fix a clock period and we go back and design to meet this clock period requirement so, we cannot habitably kind of reduce the clock period. So, our aim is suppose to increase the tr, because we find the probability is till high so, we want to reduce it. So, what to do next the like tclock cannot be reduce okay sorry tclock cannot be increase. That will affect the performance. So, the next question is that and the set up time.

It all this depends on the technology working with suppose you have working 22 nanometer technology this set up time is fix for flip-flop nothing can be done about it okay. So, we may not to able to do anything with this so, the question is that can we make reduce this combination delay okay say 0 can be make it 0 okay. So, look go back and look at the single state synchroniser we are asking whether we can make it 0.

So, no magic we cannot do any magic, the problem is that this flip-flop output is going through the combination circuit and reaching here. So, to make it 0 what we do is that we block it okay, that means we put another flip-flop here okay. That means this flip-flop output is going to a yet another flip-flop input, advantage is that there is no combination circuit in between so, as for as that batch is concern, the combination circuit is not there.

(Refer Slide Time: 47:28)



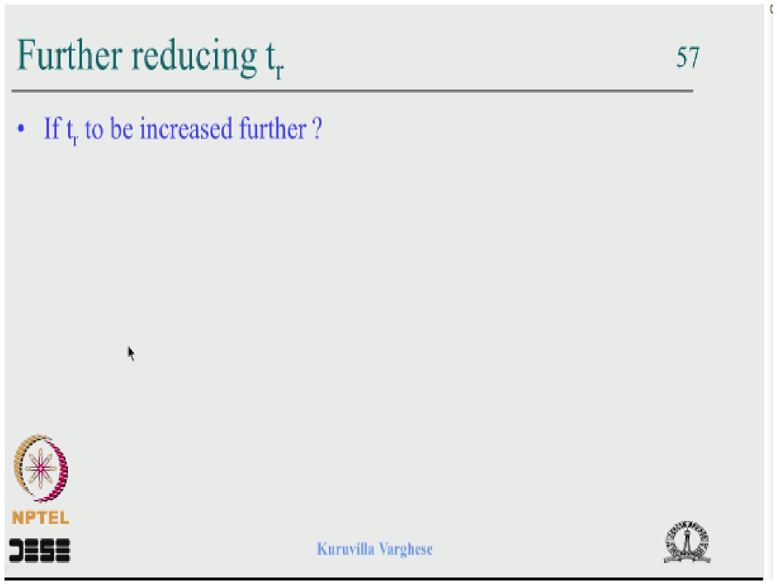
- $t_r = t_{clk} - t_s$
- Latency of two clock period

So, that is called a double state synchroniser okay. That what we do a double state synchroniser our aim is to reduce the probability of once in get into meta stability decrease the probability that would remaining in meta stability, so give more time. So, now you see that there is a clock period here between these 2, and this has to resolve setup time before the clock period.

So, the resolution time meta stability resolution time of the first flip-flop in a double state synchroniser is now increase. It is a tclock–tsetup, earlier it was only one stage, it was t clock–tcomb–tsetup okay. Now we have reduced it and nothing comes free. We know that an input changes it will take 2 clock cycle to propagate it here. So, there is latency after the input changes after 2 clock period only the input will reach the sequential circuit.

So, that is a price we take, we have a latency of 2 clock period okay. Now suppose you find the Tr now, it is better than the earlier, you find the probability, you find oh no, it is still kind of probability is quite high you want still reduce.

(Refer Slide Time: 48:54)



Further reducing t_r

57

- If t_r to be increased further ?

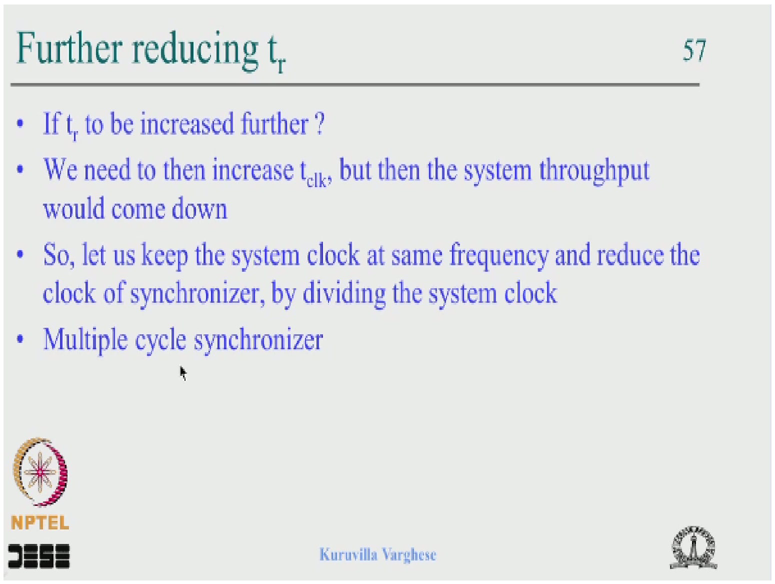
NPTEL

Kuruvilla Varghese

So, we have to increase Tr further okay. Now we are already at the tether end we have only 2 terms. One is t clock and t setup, the game is to reduce these setup, as I said this is fixed and we cannot do anything. And now we have forced to increase the clock period. But if you increase the clock period, the performance of this sequential circuit will suffer, we want to do, we do not want to do it.

So, now the question then to ask is that let us not reduce the frequency of this clock. But let us reduce the frequency of this clock okay. So, what we do is that, we will put a clock divider like assume that the clock is coming from this side. We put a clock divider say divide by 2 or divide by 3 or 4 like that, and then give it a lock clock to the synchroniser. But the sequential circuit will work with the original clock.

(Refer Slide Time: 49:58)



Further reducing t_r

57

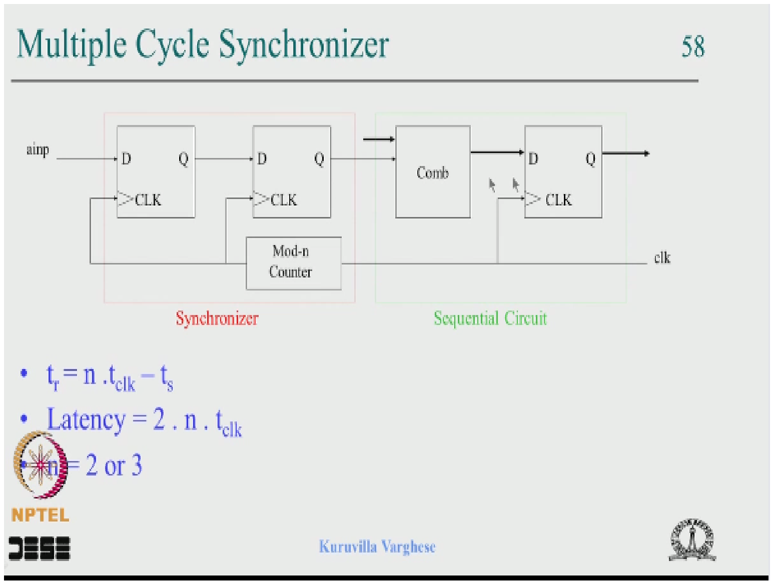
- If t_r to be increased further ?
- We need to then increase t_{clk}, but then the system throughput would come down
- So, let us keep the system clock at same frequency and reduce the clock of synchronizer, by dividing the system clock
- Multiple cycle synchronizer

NPTEL

Kuruvilla Varghese

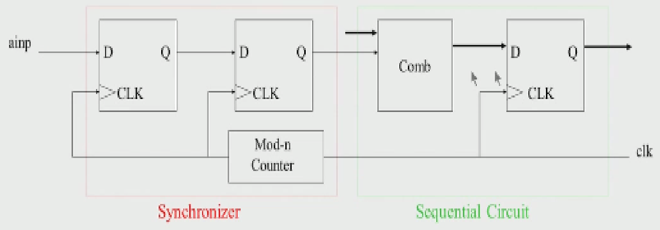
So, that is how that you know if we increase t clock system through put will we affected. So, let us keep the system clock as same and we reduce the clock frequency by dividing the system clock. And that is called multiple cycle synchroniser. Because it takes multiple clock cycle to synchronise okay, that is why it is called multiple cycle. Earlier it is double state it will wait 2 clock cycle to synchronise here it is multiple clock cycle to synchronize.

(Refer Slide Time: 50:26)



Multiple Cycle Synchronizer

58



- $t_r = n \cdot t_{clk} - t_s$
- Latency = $2 \cdot n \cdot t_{clk}$
- n = 2 or 3

NPTEL

Kuruvilla Varghese

So, that is the game here in a multiple cycle synchronizer, wherein you have the clock of the system could be sequential circuit or a data path, which is divided by a counter, mind you when I say Mod-n counter n is not a huge number like say 20 or 64 or 128. This Mod-n is like 2, 3, 4 like that you know it is not a big counter, we cannot have such a kind of latch discrepancy between the synchroniser and the system clock.

So, now you see the Tr is now, because the this was the clock period when you divide by Mod-n. The clock period by increase by n okay, it divide by 2 the clock is divide by 2, so, the clock period will be 2 times tclock. So, when a modern counter the clock period is n in duty clock so, the tr is nothing but n into t clock–set up time. So, now we having like improved the tr by order of like if it is 2, 2 times, if it is 3, 3 times or 4 times and things like that.

So, we having we have really improved, and this should be sufficient like if you find the probability of a double state synchroniser is that offer is not good enough. If you give a 2 times or 3 times it will improve. Because we are talking about exponential okay, we say the probability of that remaining in metastability decreases exponentially with time. So, if you give twice, it goes exponential and the advantage you get a exponential.

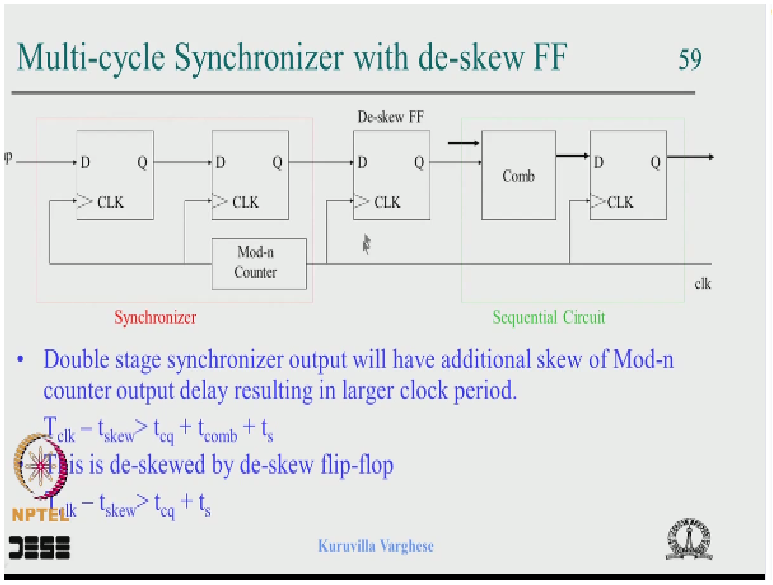
So, normally this should be sufficient okay so, and now you know that the latency is 2 clock period by, but the clock period itself is n into t clock. So, it is 2 n t clock there will lot of latency here, like you something happens, it takes if suppose n is 3 takes 6 clock cycle reach here, which we have to see whether it is kind of tolerable for the system, you have working with okay.

So, it is a multiple cycle synchroniser, which reduces the probability of this particular flip-flop remaining in meta stability for long. I know like meta this remaining in meta stability okay so, n is 2 or 3 and there is many a times people inserted of doing like this using a counter. Because at the end of it like, when you say suppose you say there is a mode 3 counter or mode 4 counter it is 2 flip-flop okay.

So, use 2 flip-flop for synchronising, and 2 flip-flop for counter some people okay now there is another issue here which we have not kind of address. You see there is a because of this modern counter there is a skew between this clock and this clock okay. Now you see that this clock is coming with the delay of this counter okay. So, and this clock comes early so, like if you analyse this path to this path.

The available time is not tclock, tclock–tskew so, it will like there is this skew affects very badly, because it each into the time available, because now it is not tclock is greater than tcq+tcom+tsetup. This tclock–Q should be greater than this particular path delay.

(Refer Slide Time: 54:25)



Multi-cycle Synchronizer with de-skew FF

59

- Double stage synchronizer output will have additional skew of Mod-n counter output delay resulting in larger clock period.

$T_{clk} - t_{skew} > t_{cq} + t_{comb} + t_s$

This is de-skewed by de-skew flip-flop

$T_{clk} - t_{skew} > t_{cq} + t_s$

Kuruvilla Varghese

So, normally what is done is that, since that to avoid that combinational the large path with this Q happening a tcq flip-flop is put. So, that the problem comes here that means you say tclock–tcq should be greater than tcq+tsetup to kind of not to affect the large delay path. So, **a** a tcq flip-flop is added here in the multiple cycle synchroniser. So, I think we have coming to the end of the lecture.

So, what we have looked at is the problem of meta stability in flip-flop. That is stem from the an issue in the cross couple latch which is used in the edge triggered flip-flop. So, we have make to setup an hold time and I said a clearly explain in somewhat detail that it comes from the master latch the setup time with some clock you know clock path inverted delay you have to subtract it.

Then that delay of the clock path will add to the hold time, then the slave latch delay is a propagation delay. And we say that for a successful operation it has to be met. If it not met 3 things can happen the output can be valid, so it can be 0 or 1 we are not sure it can be it can take long time resolve it, it can get stuck in between and this happens particularly in a sequential circuit or a data path with asynchronous input.

Because all register to register path we do the proper analysis for clock period and the hold time violation. So, nothing can happen go wrong, but the asynchronous input can create trouble, because it is not synchronous with the clock you can change anytime. And asynchronous means any input which is not having a face relationship to the clock, we are using for the sequential circuit the solution is to use a synchroniser.

That can be achieve by a single flip-flop, but if it that can get into meta stability that is better than many flip-flops get into meta stability and our hope is that it comes out of meta stability by the next clock edge. And we also see that the more time it has for to resolve. More the probability of it is coming out of the metastability. So, idea is to give more time for it resolve.

So, we have seen the metastability resolution time for a single state flip-flop it is tclock–tcomb–tsetup to remove tcomb it could 2 stages of synchroniser. Then it becomes tclock–tsetup time and if that is not we reduce the clock of the synchroniser by dividing it. Then we have n tclock– setup as the resolution time, the latency is 2 n tclock. And this counter can add skew in the path the clock path.

That can affect the total clock period available. So, sometime we add a tcq flip-flop, so that is about the synchronization. So, I have address issue, I have already given a solution, so what is remaining is maybe little more detail 1 or one more solution and something specific to the state machine when the input is asynchronous and something about the reset, then we can wind it up and hoping that in the next lecture half a through we can wind it up.

So, this is an important topic I was doubting whether I will be able to cover that in the course. But then not to talk about is kind of is not good, after learning you know so much about how to design and all that. so, I think it is an important topic please go back read it and if you want additional material you can kind of look at the textbooks if it is available or look at some papers the white papers which is available in the with manufactures. So, that this can be learn well, so I wish you all the best and thank you.