

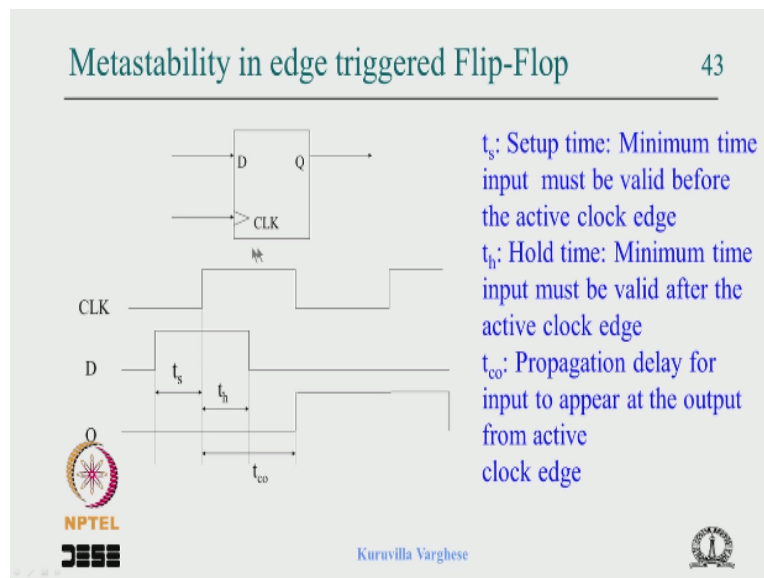
Digital Systems Design with PLDs and FPGAs
Kuruvilla Varghese
Department of Electronic Systems Engineering
Indian Institute of Science-Bangalore

Lecture-30
Synchronization 2

So, welcome to this lecture on advance digital system design the course digital system design with PLDs and FPGAs. The last lecture we have started with the effect of metastability in sequential circuit and data path. We could not we did not spend much time on how the metastability happens in an in a flip-flop or latch or anything like that though briefly I mention the cause of it in a gist. The essentially our focus was to look at the effect of meta stability in sequential circuit and data path.

And how to handle that situation. So, that proper operation of the circuit is not impaired definitely analysing the metastability in detail starting with the cross couple latch moving to kind of edge to get flip-flop and all will bring clarity. But due to lack of time we will set assident focus on what is useful for the design of the digital system, we will handle it, that is a plan. So, let us run through the slides we have looked at last lecture.

(Refer Slide Time: 01:48)



So, we know that in an edge triggered flip-flop for the data to appear at the output from the input. The input data has to satisfy some timing requirement with respect to the clock edge active clock

edge. The data has to arrive sometime before that is called setup time. Not only that it has to remain at the input, after the clock edge for a while. That time is called the duration is called hold time, if that happens then from the clock edge with a delay the output appears okay.

And this I said if you analyse a master slave flip-flop which from which this edge to get flip-flop is constructed during the negative for a positive edge to get flip-flop. During the negative clock period the master is in transparent mode. So, the setup time would normally mean from the active clock edge that is from when the master is going to be cut off from the input actually the minimum time required for that latch to set.

Because that is when the latest the input can change if afterwards the input change, there is not enough time to set the master latch. So, that is what is the setup time of course normally in the clock path there will be invertors to choose a correct polarity and to choose the opposite edge for the slave and all that. So, that normally there will be 2 invertors, so that invertors Q has to be removed from the setup and the hold time is that inverted delay itself.

Because even after clock going high it take that inverted delay for the master to cut off. That is a whole, so during that time the data should not change. And when the positive clock edge comes, so the slave get the copy of the masters, slave is an enable, the master is disable. So, for the time for slave to set is the propagation delay, normally the again there will be an inverter that delay has to be added with latch delay, slave latch delay for propagation delay.

So, that is a essence of this setup hold and propagation delay , but again in a basic course many a times you have told that, if this is met the input is captured at the output kind of truthfully. But many a times it is not discussed what happens, if it is violated okay. So many things happen as we said if the setup and hold time is violated kind of the flip-flop can miss trigger that means, suppose the previous value was 0 and it was trying to set to 1, it may remain in 0 or it may kind of set to 1 okay.

Now so, it means that you are not really sure what the output is going to be, it could be 0 or 1. But as I said normally it is not a dangerous situation as long as input purses for at least 2 clock

periods, then definitely in the next clock edge everything will be okay. Because it will meet the setup and hold time and then the data is captured at the output properly okay. So, miss triggering itself it is not a very serious issue. But the second effect is that it might take longer time to resolve it.

That means if the data changes close to the active clock edge, it will long time for it to settle. And that is dangerous, because a like we have in a sequential circuit, we find the minimum clock period and once you decide that if the data the output takes longer to settle, then it can violate the setup and hold time at destination register not no issue with this flip-flop. But it is driving something and that destination register it can upset the setup hold time.

And the worst effect would be this output can gets stuck to a voltage level which is not neither one nor 0. It will be in between and that can drive the any circuit not even flip-flop, it can even drive the combinational circuit into active region. The gates and that will act as amplifier any slight noise, it will start switching, it will dissipate lot of power maybe it is taken care in the design depending on how you design.

But definitely this is quite disasters that it get stuck in between and the worst thing is that we cannot definitely say when it will come out of it we can only say about the probability. So, that means it can remain there for a long time, when we talk about probability that does not kind of preclude chances of it, getting stuck for a say 5 second, 5 minutes or anything like that, though we talk about the probability, you should remember that.

But that happening often when the probability is low may not happen. But then like the probability is probability, so you should not have kind of a deterministic view on the probabilistic events, that should be kept in mind.

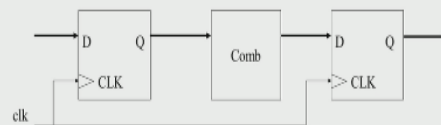
(Refer Slide Time: 08:01)

- 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.



So, let us come to that the effect of it, but in a sequential circuit or in a data path we take care of all these, we like we choose the proper clock period to meet setup time. We analyse the effect of this Q, we analyse the effect of hold time. And this Q effect on both we will consider, though we have not discuss that in the in our lectures. But we will be doing that, when I take the FPGA part of the course.

(Refer Slide Time: 08:27)



Min Clock period / Max frequency

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

Avoid Hold time violation

$$t_{co(min)} + t_{comb(min)} > t_{h(min)}$$

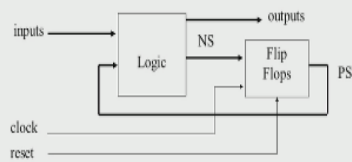


So, in a data path we do the maximum frequency analysis, hold time violation.

(Refer Slide Time: 08:34)

Sequential Circuit: FSM

47



Min Clock period / Max frequency

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

Avoid Hold time violation

$$t_{\text{co}(\min)} + t_{\text{comb}(\min)} > t_{\text{h}(\min)}$$



Kuruvilla Varghese



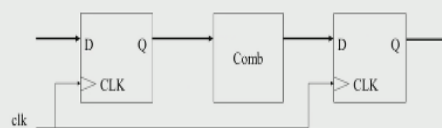
Similarly in a FSM we do both.

(Refer Slide Time: 08:37)

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?

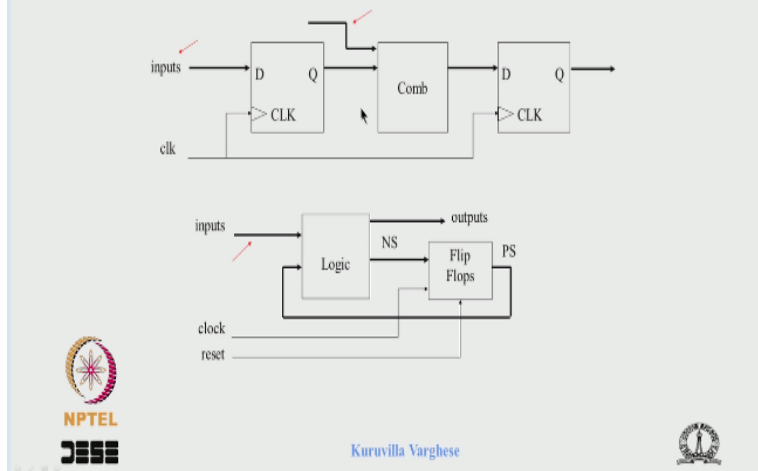


Kuruvilla Varghese



And then when the metastability can happen in this circuit is a question.

(Refer Slide Time: 08:45)



We said yes there could be inputs directly coming to this combinational circuit in a data path. There could be inputs coming to the first stage of the data path maybe this continuous you know. There is register combination circuit registers, combination circuit and so on. But at some point, somewhere some data from the external world or the data path comes in. And in an FSM there are inputs mainly this is from data path.

But we are not sure the frequency of the clock from where this input is coming okay, it may not be same. So, or it could be a some process, which is generating a logic value like a limits switch , which is not relation with the clock many a times. And then these inputs may not meet the setup and hold time as for as this input is concern may not meet the setup, and hold time and as for as this is concern it may not meet the setup and hold time.

This latency as I said it is a matter of shifting it does not affect the probability of any particular input you know creating metastability in this flip-flop. Similarly about this, so the problem as for as the sequential circuit, or a data path is concern, the asynchronous input is the one which can cause metastability. So, once again though we have metastability our concentration as for as the sequential circuit and data path is concern should be on inputs which are not synchronous okay.


So, if you are a good designer re you should focus your attention. The inputs coming from an asynchronous domain okay, that should be handled properly. That is where cracks of problem okay. That is like asynchronous inputs.

(Refer Slide Time: 10:51)


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



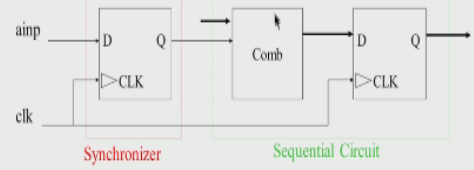
Kuruvilla Varghese



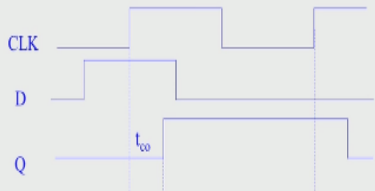
So, when we say asynchronous input it will be an input which is generated as a on a flip-flop or a register with different clock or a process, which is not synchronize to the sequential circuit. So, very easy to answer this question, that problem is that the input is asynchronous. So, solution is to synchronize it, and it is again we do not know probably at the beginning you do not know how to synchronize.

(Refer Slide Time: 11:20)


Single Stage Synchronizer 51




Synchronizer Sequential Circuit



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



Kuruvilla Varghese



But it is very easy, suppose there was an asynchronous input which is going to this combination circuit and reaching a destination register, the way to synchronise is that you connect that input asynchronous input to a D flip-flop, clock by the same clock as a sequential circuit and the Q is connected here. Now what happens is that the D is captured by this and appears with a delay. Now anytime D changes.

This is captured at the Q with the delay of a propagation delay T_{co} , and that you know transmit through the propagate through the combinational circuit reach here. So, if the clock period is chosen properly in the next positive edge here, this input that means the output of this synchronising flip-flop will meet the setup time and hold time, there should not be a problem.

And of course the input which is synchronize will reach at this destination register with the latency or delay of one clock period. So, that is a penalty we pay penalty in terms of time re is a latency and penalty in terms of the area which is a flip-flop okay. So, nothing comes free so, you suffer in delay and suffer in the area okay that you should know.

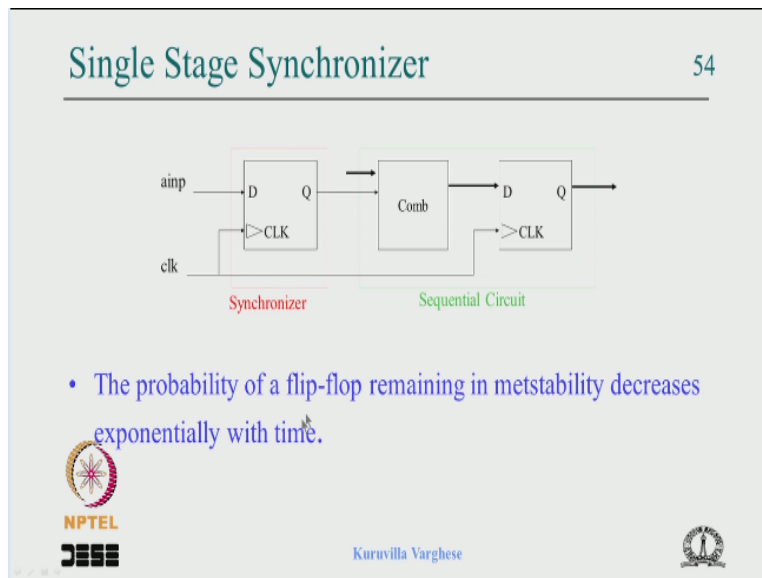
Now as I said, we have not solve the problem in a very kind of fool proof way, earlier there were chances of all these or some of these flip-flops getting into metastability. Now there is no guarantee that this will meet the setup and hold time of this flip-flop. Because, this is asynchronous with respect to this clock. So, this can get into metastability okay now that is better than the earlier case like we are kind of sampling the asynchronous input in 1 point.

But earlier it was sampled by different flip-flops and maybe some will get into metastability some will not, some will miss trigger it was a mess okay. It was you could not say what is the kind of what is happening at the output you know. Some could be 1, some could be 0, some could be in metastability and so on okay, though the way I say you might think that it occurs so frequently does not okay .

But you know that I want to highlight the problem of sampling an asynchronous input at multiple places with different path delay okay. That is a issue. But our hope is that if this synchronizing flip-flop get into metastability, by the next clock edge if it comes out of it and if there is enough

time for it propagate through combination circuit and meet the setup time at the destination register it is okay.

(Refer Slide Time: 14:43)



So, that is the real hope, but like this I want to state I can only kind of make a statement, we have not we are not going to analytical part of it, the fact is that the probability of a flip-flop remaining in meta stability decreases exponentially with time. That means it just synchronising flip-flop get into meta stability as the time passes, the chances of it coming out becoming exponentially more.

So, every nanosecond you give extra, it is an exponential increase in the probability of it coming out of meta stability, again I warn you it is a probability I mean if not certain like you cannot say after you give 1 nanosecond extra. It is surely now with the double chance it is going to come out it may remain there okay. There is, what there is a kind of logical problem in the last sentence.

But I hope you got the picture so, idea is that we give enough time for synchronising flip-flop to come out of the metastability. And which from now onwards we can analyse that part the time available for it to come out of metastability is that. This path there is a clock period 1 clock period a clock comes, and before the next clock edge this output should reach here. But now if it get into metastability.

It should come out propagate, and meet the set up time here okay so, how much time is available for to ~~to~~ come out of meta stability, that is the total clock period – this time for the propagating through the combination circuit–the set up time, at least by then it should come out okay. Suppose the clock period is 10 nanosecond, set up time is 1 nanosecond and the combinational delay for 4 nanosecond. So, the put together it is 5 nanosecond.

So, we are hoping that within 5 nanosecond after getting into metastability this comes out of it then there no issue everything goes clean. But that also tells us that if you are able to do give more time like then there are more chances of it. It coming out of metastability so, it is a question of analysing it finding the probability. And checking is that okay not only the probability maybe.



We have to convert that into the write of metastable kind of failure metastability failure in the sense that does not come out of metastability, or the average time between such failures okay. So, if it is large enough and if it toggles very seldom then it is okay.

(Refer Slide Time: 17:47)

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

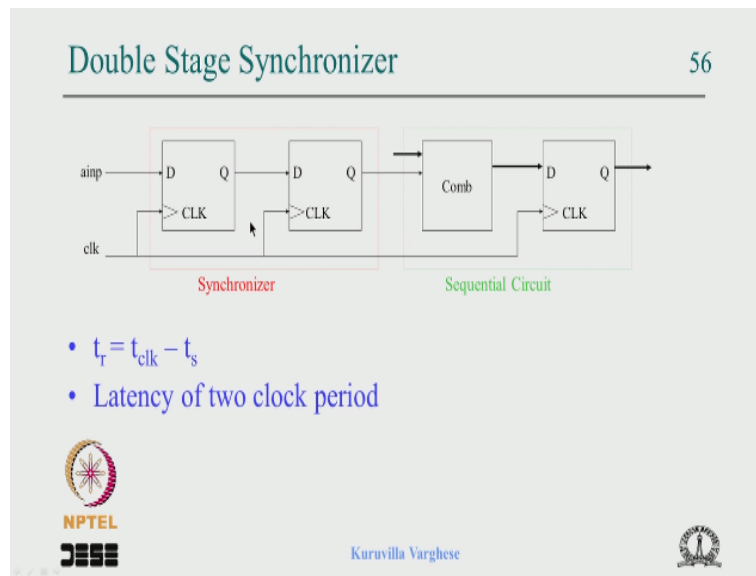


Kuruvilla Varghese

So, but a if that is not okay, like you have the meta stability resolution time for a single state is $t_{clock} - t_{comb} - t_{setup}$ then we have to increase it we cannot kind of at least from the at the outset. We should not tamper with the clock period if you reduce it the system will suffer so, we reduce this combination circuit and we are trying to make it 0, and that is by you know that by putting

another register in a chain here. So, that the output of this flip-flops is only a flip-flop not a combination circuit so, that is double state synchroniser.

(Refer Slide Time: 18:25)



so, we put two flip-flop now time available for this to come out of metastability is clock period – a setup time okay. But price we pay is the 2 latency, 2 clock period latency. So, if something happens here, the worst case it will appear here after 2 clock period okay. That is a and of course in terms of area to kind of flip-flops. Now suppose if like you know kind of assess the probability.

And find that it is not enough meantime between failure it is not very large. Then we will be forced to kind of increase the clock period. But then this should not be affected. So, what we do is that we divide the clock for this part and retain the original clock for this.

(Refer Slide Time: 19:22)

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



Kuruvilla Varghese

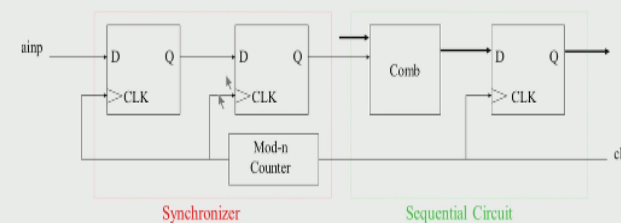


Sequential circuit or the data path main data path.

(Refer Slide Time: 19:25)

Multiple Cycle Synchronizer

58



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



Kuruvilla Varghese



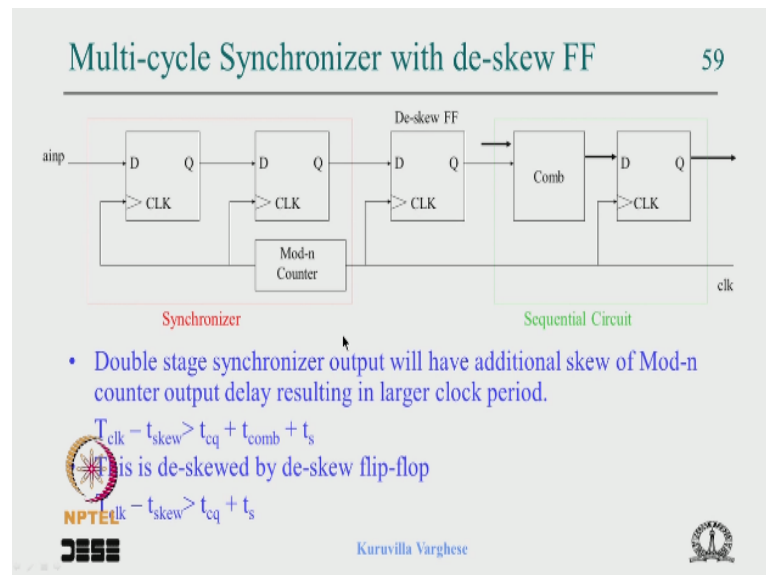
So, that is what is shown here, you divide by n the clock is coming from here. The main clock goes to the sequential circuit divided clock goes to the synchronizer and now we have clock period is n into T_{clock} . Because we have dividing by n , then the clock period increases by n —setup time. So, that should normally solve the problem n is and now the latency is 2 clock period of the divided clock which is $n t_{clock}$.

So, 2 and t_{clock} is a latency which can be high. But n itself is 2, 3 or 4 things like that not a huge number. But I said there is an issue here of a Q. Because, there is a Q between this clock

and this clock. This the source registers clock is a delayed version of the destination register clock. So, like if you put the destination clock like that and the source clock is coming delayed okay.

So, that into the available clock period, so available clock period for the propagation will be $t_{\text{clock}} - t_{\text{Q}}$. So, this Q will force one to increase the clock period okay, that is not a good idea, and we will definitely analyse the effect of Q in the clock period all that. In the hold time violation, and all that in different scenarios you know there could be different scenario depending on where which side the clock comes from. So, we will analyse all that, but at least I want to tell you the problem here.

(Refer Slide Time: 21:09)



So, what we do is that to avoid that, we put a De skew flip-flop. Because, now the skew come between 2 adjacent flip-flop without any combination circuit. So, it is kind of you know $T_{\text{clock}} - Q$ should accommodate the t_{cq} and the t_{setup} which will be much smaller than this will be much smaller than this which is like this. Suppose if there is a skew here, the situation is going to eat into the clock period.

And it is going to push the clock period up okay. So, this would not and that is the de skew flip-flop definitely it will add latency to the old process now. So, it will be the latency you can work

out it is 2 and $t_{clock} + t_{clock}$, because is this not clock by this kind of output of counter. But the input of the counter.

So, that is what we have looked at the last class. I have repeated it, because it is important that should get into your mind very clearly. And this is a kind of standard solution almost people treat it as a kind of thumb rule, wherever they see an asynchronous input. They put 2 flip-flop in a chain, sometime they 3 flip-flop okay, we will see why 3 flip-flop.

I do not quite agree with such kind of thumb rule like which is like many a times, if you ask some people like you sampling an input say the nickel street how many times you should sample. They will say 10 this 10 is not a magical number you know that we have 10 fingers in the hand. So, we and we use a decimal system maybe because we have the 10 fingers in the hand, because earlier the counting was you know.

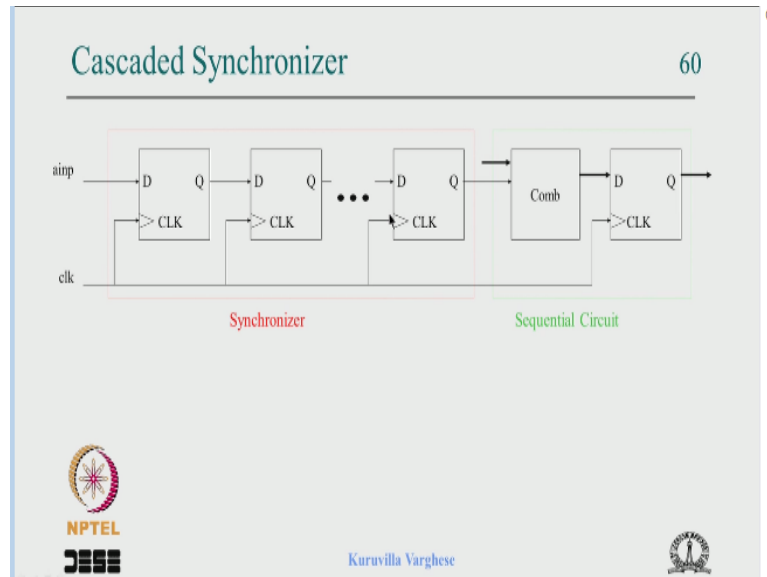
Based on fingers and we will keep a tally for how many tens and that is how the decimal system, you know kind of arouse and before that people will like even for now for the time we use 12 as a base. So, there were like d or decimal number system was there. So, what I am talking is that my argument is that, if you had 12 fingers then we will be all the time telling 12 rounding of to 12 everything.

So, just relying on thumb rule for design is a very bad thing, and now a days, anyway it does not work, because people want very optimise circuits in terms of very power dissipation delay and so on. But you should as a designer you should not resort to these kind of in the magical tricks which is told by other people to do that whenever you see an asynchronous input 2 flip-flops or 3 flip-flops without any questions asked not a good idea.

At least when you are working in a see most technology, you know kind of the delay is you can expect in that technology and from there you go back and analyse it is the data is difficult to come by for metastability analysis. But I am hoping that you get it at least look at some the publish work on that and you know assure yourself that the scheme you put is enough, or it is not kind of you are not over designing and so on.

Like where you the 2 double stage is enough do not multiple stage cycle, multiple cycle synchroniser and so on. So, let us come back to it.

(Refer Slide Time: 25:01)



Instead of multiple cycle synchronizer, sometime people put a kind of cascaded synchronizer, that means they will put 3 flip-flop in a chain, 4 flip-flop in a chain so on the argument is that like say you can work out probability like if you know frequency of the clock. And the set up and hold time into width, and if you know the rate of change of the asynchronous input, because this need not be kind of regular square wave.

So, the average 0 or 8 are the maximum write whatever depending on average write should be good enough to take. And then you can find out the rate of metastability entry, then knowing the time that the fact it will as you give more time probability if it coming out increases exponentially with time. So, the probability of it remaining the decreases exponentially with time okay so, from there.

We can find out what is rate at which the second flip-flop can get into metastability. And again do the same analysis for the second flip-flop, then you know what is the rate at which the third is kind of getting, but this you know that the probability gets multiply and the chances are

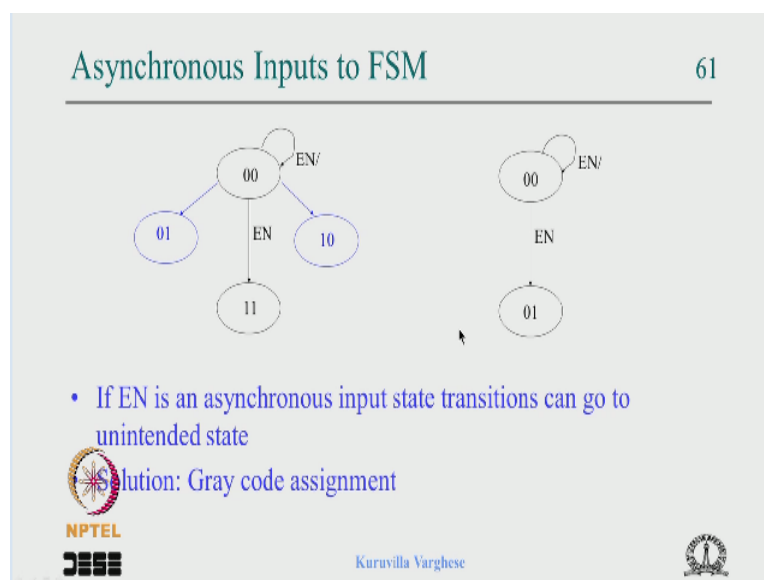
becoming less and less for you know when you come towards the real sequential circuit the probability of that the last flip-flop can getting into metastability is very less.

So, the mean time between failure will be a huge numbers in terms of maybe millions of years that should be good enough for large number of products which should make out of that design okay. So, that is the basic idea about the cascaded synchroniser the probability of meta stable output reduces multiply that means that remaining in metastability reduces $(\frac{1}{2})^{(26:56)}$ at each stage of the cascaded synchroniser.

So, some people do this instead of, and it does not matter you know it will be very frank like here like if you assume 4 flip-flops okay. So, you have a latency of 4 here okay, and you are using kind of 4 flip-flops. And if you go back to this, and you put a divide by 4 counter more 4counter, then use 2 flip-flop the latency is 2 and t_{clock} , which is nothing but a 4 clock cycle of this.

So, essentially kind of things are you know similar there no difference it is the matter of kind of some choice only thing is that maybe this introduces as q . But in the cascaded there is no skew so, many a times when you see some kind of solutions at least orbitically some time it all means the same. Though the argument the structure is different, so one should look at it and try to make sense out of it okay.

(Refer Slide Time: 28:08)



Now let us look at the effect of asynchronous to input FSM, now I am was warn you saying that there is no reason for you not to synchronise and asynchronies input in a very serious design maybe this concern is little bit too old, where one use to use kind of discrete kind of devices, where a flip-flop can be very costly maybe putting 2 flip-flop will end up you will end up using a kind of 20 pin not 24 pin d package which occupies.

But now a days SOC chips are you know it shows small then the so much density of transistors or gates not a problem not much of a problem. But then it is nice to analyse at least academically will learn something in a complex situation what is a effect of the asynchronous input. That is the only thing so, I would suggest that if there is an asynchronous input synchronise it rather than applying this technique, which I am going to say.

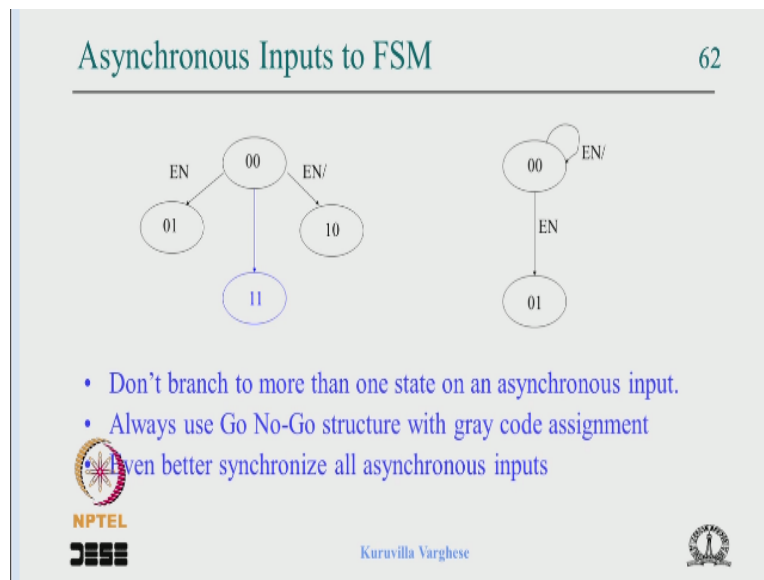
So, assume that this enable is an asynchronous input to a state machine, and this state machine is like that as long as it is slow remain there. If it is high transit to the state with 11, now mind you there are 2 flip-flops q1 and q0. And now the situation is that we are not synchronising, this asynchronous input is going to the input of q1, that flip-flop and the 0th flip-flop okay. Now because of the path delays difference and the path delays.

This will be sample at different point in time, and you can imagine sometime it may happen that the q1 transit to 1, and q0 remain there. Because it is asynchronous, or the opposite can happen the q1 remain in 0 that means miss triggered it retain the previous value. But the q0 as transited so, it may happen that instead of the valid state 11, when the clock comes it transit to an kind of invalid you know state 10 or 01.

And we do not know about the state, and this could be kind of you know from state within the state diagram, or it could be unused state, whatever it is dangerous. This is not like output rises where it goes briefly there, and come back it really miss trigger. So, you can imagine that you have a very complex state machine which loops and branch an all that. Suddenly you find that it is in some state and soon you find that the state machine is elsewhere and it continues from there and it is quite dangerous.

So, you should never suppose we not synchronising the solution is not to do such a kind of assignment, you should make it grey code. So, that only one of the flip-flop change so, even if there is a miss trigger does not matter, either it will remain 00 or it will transit to 01. So, always do a go, or no go like enable bar remain there enable go there. But this should be grey code if you are using an asynchronous input.

(Refer Slide Time: 31:50)



The similar situation can happen, when the state machine is in say 00, and enable is an asynchronous input. When it is 1 you transit to 01, when it is 0 the machine transit to 10 okay. state machine transit to 10, now since once again since it is we are talking about in this condition when enable is 1 Q0 is changing the state, when enable is 0 Q1 is changing the state. Now you can imagine that this 2 flip-flops sampling 1 input with different path delays.

So, it may happen that 1 instead of both changing 1 might change. So, you are end up you are hoping like here sorry here only Q1 was changing here only the Q0 was changing. But Q1 will kind of interrupt it has 1 and it will change, and Q0, so will interrupted it as 1 and change. Then you will end up in another invalid state, and this is quite dangerous again the solution is that never branch on asynchronous input like this.

Like when it is 1 go to some state, and 0 goes to other state, you should always adopt a go, no go structure, when it is 1 transit, when it is 0 remain there, and this should be grey coded. And as I

said always it is even better to synchronise the asynchronous input without much botheration, unless it is a very small circuit here trying to suppose you have making a 4 bit counter.

And you have an up down input, direction input from somewhere. And, if you double synchronize the up down and direction, then that will end up using 4 flip-flop which is equal to counter flip-flop. So, but this situations are rare, and if at all use a 4 bit counter it will be a very small application. So, that you should kind of remember, so better to synchronize asynchronous input related to it.

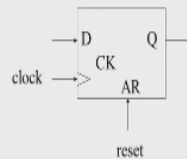
So, now essentially what we have covered is how to handle what is metastability, what is it effect on a flip-flop. And in a sequential circuit and data path how it affects, and how to I mean where the problem really come in a sequential circuit or in a data path. And we said the problem is asynchronous input. The solution is to synchronize it, so we have looked at the single state synchronizer .

Double state synchronizer which improves reliability or probability aspect is improved. Then multiple cycle synchronizer are cascade synchronizer. And we have seen what happens, if you use asynchronous input in an FSM to make branching okay. So, we have 1 branching with multiple flip-flops changing at the same time. And, 1 branching where depending on that input, it make a decision to go to one side or the other side. So, that is what we have seen.

(Refer Slide Time: 35:36)

Reset Recovery Time (t_{REC} , t_{RR})

63



- Reset Recovery time is the minimum amount of time between the de-assertion of reset and the next rising clock edge, for the proper sampling of 'D' input Flip-flop.



NPTEL

Kuruvilla Varghese



Now a related kind of scenario is this something called reset recovery time okay. That is related to the asynchronous reset in a flip-flop, which is a kind of this recovery time, the notation is t_{REC} like t_{REC} is the recovery time or t_{RR} okay, the RR and REC is subscript. So, essentially this unit kind of realise the learn nevertheless it is a very important aspect.

You would have assume that at least the behaviour of an asynchronous reset in a flip-flop is that you assert that you make it active with the delay. The Q goes to 0, and you remove it, then clock comes everything works fine is what we assume okay. But there is a factor called recovery time, that means that the reset recovery time is a minimum amount of time between the D assertion of this signal.

And the next clock edge active clock edge. Suppose the clock is coming the positive clock edge is coming here. The reset should be removed sometime before that okay. Otherwise this output can get into metastability okay, and this is quite serious. Because in a digital circuit there could be tens of thousands of flip-flops okay, or hundreds of thousands of flip-flops.

And many a times at the power on a single reset can reset all the flip-flops okay, most of the flip-flops and it has to be reset. Because precisely same reason there is no guarantee that if you power on all the flip-flops all the outputs are going to be neat you know like during the power on

probably it can get into metastability. So, reset is essential, but if you do not meet the reset recovery time in flip-flop.

Again the metastability can happen in the flip-flop. So, this is something which is overlooked, and many a times one think of the assertion of reset not the de-assertion that is a problem like we are not asking the right question, when you assert it fine reset is driving the internal latches, you know that is the problem. Like in an edge triggered flip-flop you have a master latch. And a slave latch as I said.

That and this reset the asynchronous reset is directly going to the master latch, and slave latch and resetting it. And you can look at circuit diagram for it. Then you will realise why that happens okay. Again in this course, I do not have time to go into the details of that, because it will eat up into our time. So, when that means they are driving the master and slave latch directly not going through this clock gating you know, that is why it is asynchronous.

As soon as you assert with the delay this is reset, the output is reset. The clock does not matter, because is going directly into the latch, and what the clock does is that it gates the input to the master latch so on. So, we have by-passing that. But now when you remove it, it may happen that you are kind of this is being input is driving the master latch, and it is nearing that setup time.

It is close to that minimum time required for it a setup, and you have missing it with it through the reset it is trying to set through D and we are removing it, we have driven into 0, but we are removing it very close to the clock edge. So, then this will not have enough time to set it up. So, this should be and how much time is not the setup time, because this is directly going to the latch.

So, this is the kind of reset recovery is that time delay of that path where this reset is going and kind of resetting the latch okay, that path. Like if you analyse the detail circuit you can find reset recovery time in terms of the gates involved and so on okay. But I hope you get a picture of a

situation okay, that is a problem like you can assert it anytime with the delay it will be kind of reset.

Because it directly going it, does not matter, but when you remove it, and it is near the positive clock edge, near the setup time. Then this thing can get into metastability, because 2 drives are there. And this does not of enough time to drive it to the proper value. So, this can get into met stability so, this has to be met, and that is now so you would see that most people.

Most vendors designers would advice, you to use synchronous reset because it is safe okay. But there is an like nice feeding about asynchronous reset saying that like a outputs the flip-flops are reset without a one clock period delay. Because in the case of synchronous reset there could be a the worst case one clock period delay for it to reset okay. So, if you want to reset something very quickly.

Even before that for whatever reason maybe for kind of reliability fault tolerance or critical situation whatever this maybe good so, now understanding the reset recovery what is the solution okay. Now if you say kind of okay let us synchronise the asynchronous reset then it is like synchronous reset only. Because what synchronous reset do we do it outside it really it meet the kind of reset recovery time.

But then it is a synchronous reset the behaviour is synchronous, because you assert reset it comes with the one clock period delay over case one clock period delay it is like synchronous reset. But we can say this with certainty so let us not synchronise leading edge. But let us synchronise it trailing edge, because if you do not synchronise the leading edge as soon as you asserted the reset happens.

And the problem is about removing it near to the clock edge. So if you synchronise to the clock edge, it is removed after the clock edge. And it will definitely meet the recovery time at the next clock edge so that is the idea.

(Refer Slide Time: 43:12)

Reset Recovery Time (t_{REC} , t_{RR})

64

- One way to meet t_{RR} is to synchronize asynchronous reset input to flip-flop, then the reset behaviour would be same as that of synchronous reset, as the reset happens at the clock edge.
- To retain the asynchronous reset behaviour only the trailing edge of the asynchronous reset should be synchronized.



Kuruvilla Varghese

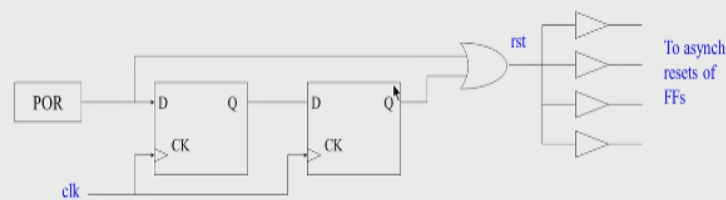


So, that is what I have written here to retain the asynchronous reset behaviour only the trailing edge of the asynchronous reset should be synchronise. So, that is the game.

(Refer Slide Time: 43:24)

Asynchronous Reset

65



Kuruvilla Varghese



So, I am showing straight away a possible kind of solution there are many other, but I am showing the again the simplest one. So, assume there is power on reset circuitry, which has know kind of relation to the clock, it comes so, if it is kind of going to the asynchronous reset of flip-flop then you can call meta stability. So, we are synchronising it so, this is the synchronise version of the reset, it is an active high reset signal as the name suggest active high.

And you can imagine that this part is the delay, the worst case 2 clock period delayed version so, now the trailing edge comes much later. Now what we are doing is that we are over ring together okay so, what happens is that leading edge come as it is, because it over with the delayed version. But the trailing edge is synchronise version so, that is the clever trick for a active high reset, now the same technique I hope you got this.

We have double state synchroniser for the power on reset so, this is a synchronise version. But both the leading and trailing edges add are synchronise, now what we do is that we kind of over the original reset. So that we get leading edge as it is okay so, it is kind of over ring of a pulse with the delayed version of the pulse. So, you get the leading edge as it is, and the trailing edge is a synchronise version which comes with the kind of worst case latency of 2 clock period okay.

Now for the active low signal that means this is kind of like it goes in the opposite direction 1 then goes 0 and come back to 1. So, we do we replace this AND OR gate with an AND gate and we have done it okay. So, it is a kind of 0, and delayed version of 0, you know that for the AND gate any input is 0, the output is 0. So, it is like a as for as the active low signal is concern. We have studied at the beginning if you remember.

We have looked at the difference between AND function and AND gate okay, now this is an AND gate doing an over function for an active low signal as so for an AND gate the inputs and the outputs are active low. Then it is doing in our function and we have discuss this in the over view, when we have kind of done a review of the basics. We have done at and there is basically I have told at that point in time.

That is bringing in the Demorgan theorem into our concept okay. So, this is the circuit for active low reset and naturally when in a check there lot of flip-flops to be kind of reset. So, it is not a good idea that 1 wire you know drive everything. You know you need to kind of buffer it properly for the because of fan out is limited. If suppose the fan out of a gate is kind of say 50. Then if there are 1000 then the unit 20 buffers.

And now you have to balance it out is not a good idea that you put a buffer then you drive 20, then again you buffer it. So, everything get delayed in a chain so, you do a balance buffering, and this goes to how ever money you want you put it parallely. And these branches drive out sub sections of the circuit that is what is showing shown here. So, essentially what we have looked at in the last few slides or the reset recovery time which is related to the metastability.

This is related to the asynchronous reset of a flip-flop, since the asynchronous reset is kind of driving internal latches directly if the no problem in driving it. But while removing it if the clock edge is kind of is near to the clock edge. Then that is the set up path set up time of the input, that can kind of interact with this drive and cause metastability in the output of that flip-flop.

So, the solution is to synchronise but then it does not have an asynchronous behaviour so, again we said it is fine if synchronise trailing edge rather than leading edge. And the solution we have looked at is that, let us synchronise it, but do an over function okay. So, in the case of an active high reset, we use an OR gate, and for the active low reset we use an AND gate. And that was solution.

So, kind of I wind this metastability synchronisation part here. So, essence id that when you find an asynchronous input you have some idea of what is mean time between the failure with regard to the meta stability or a when there is when you encounter the asynchronous input, and do the synchronisation. And we have not handle all possible ways of doing synchronisation.

But this is by forth the most useful part at least for a basic course. I think that should be good enough and of course the reset recovery, again we have discuss what to do which but this has to be applied kind of without any hesitation what so over okay. So, that you can blindly kind of replicate whatever I have told so, let us move on so, with this I have kind of I minding up the digital design part of my course.

I have given you how to do a hierarchical top down design, we have seen an example of a CPU then we have looked at what is the data path, what is the controller, what is the controller behaviour, What is the structure, how to design that structure, how to design a control algorithm

whole methodology we have seen, and we have looked at a case study where everything is applied, all these principles are applied very maybe not a very complex case study.

But it was a kind of real life case study we have looked at all practical step. Then we have looked at the various issues in the finite state machine like power on to is a clock frequency, output races. Then we have looked at the problem of state assignment, unused state. Then we have looked at how to reduce the output delay by decoding from next state logic, by encoding the output in state bits all that.

And then we also have come to this problem of metastability in flip-flop , asynchronous input, synchronising it, reset recovery. So, kind of I have covered, and I have given a kind of a review of your basic you know basics you have learn in undergraduate course that you have familiar . So, I think with this you should be able to kind of a make good design.

And we will take case study. Now we have learned VHDL we are kind of completing that, now what is remaining is the 2 devices, device technologies the programmable logic devices, and the field programmable gate array okay. Now once I complete that and bit of VHDL we are not covered the test bench. Then we can kind of look at the case studies were play with the tool we implement that on a board.

I do not have too much time to show you lot of case studies and so on. But then I will show something that should be enough. And now a days the people are kind of solving lot of problems you know like say in mathematics the people try to solve all kinds of problem. And yes that is good, because that makes you quick, and that works out in examinations where in a short period of time, you are able to solve more number problem.

But in real many a time that is not the situation, that is not that in a fraction of a second you have to solve a problem okay creatively, elegantly without much cause, without causing harm to the humans, environment and things like that okay. And so, if when you work out some example case study that should have the real life element in it. It should address all the basic issues involved.

So, that then very few case studies are enough instead of doing all lot of projects, you know there is some people resuming full of projects you know. And that we have they have done this, they have done that, but nobody ask how come in a short time, somebody is able to do so many projects and understand it well. It is better you do few of projects, few of things with a greater grasp, and kind of a come out with the creative solution, the elegant simple solution you know.

elegant, when you say elegant it is a qualitative word you know we cannot kind of put numbers in all tools as something is elegant. But then it appeals that solution appeals, something is elegant the music is elegant, the way somebody kind of dress is elegant or when you look at the nature it is kind of beautiful, and that can be applied that is a little bit qualitative aspect.

But then we are able to judge that something is elegant or not so. That is just a kind of not justifying the fact that I am not able to kind of do lot of case study I can do it, I have enough maybe I can kind of do a course full of case study, there is no issue I have so much stock with me. So, that is not a point, but that does not educate you increase your depth of knowledge and so on.

So, I am trying to balance it out you know now, we have I am trying to you know stress on the device technology, the programming language. Even with the programming language I have emphasise what it means, what this construct means and what equations it create, what logical structure it creates and so on. Rather than you know simple dwelling on the all kinds of syntax and all kinds of jugglery with the syntax no point you know.

There is for a for an expert person even a simple tool would supplies you know. You have a very good photographer is able to take a good photograph with a maybe a simpler camera. But, it given to a even the most advance camera you will not able to that. that is 2 with a any tool, so this sophistication of tool would not guarantee that what you design will be kind of efficient elegant fault tolerant and all that.

That comes with expertise, the creativity and logical thinking and systematic working and all that. So, that you should keep in mind, so my plan is now to complete this programmable devices I did not start it. Because you will be stuck in between, it might 2 or 3 lectures. So, we will handle the evolution of it. The historical evolution of it, the simple PLDs which have very rarely used nowadays.

But maybe there is a use in some cases for it and there is a complex PLD again application is limited, then we will have the 1 part of VHDL which is remaining is test bench. And there is another part of functions and procedure, if time permits I will take it. but it is 1 part of the language wherein it is very similar to a programming sequential programming language.

Of course there is something called operator overloading which is related to synthesis. But with whatever I have discuss you can kind of grasp it. Then we will go on with a FPGAs, then we will play with a tool and the case studies and that should kind of wind up you know put everything together, and give you a complete picture of designing the digital system practically.

And we will equip you to kind of start in the path of digital design. I would not say that will kind of make you an expert in the front, and VLSI design of redundant FPGA design. But that is a good starting point, so that you can continue you can learn, you can practice and do well. So, please now like we have covered almost two third part of the course. The last part is coming up, so please review whatever we have done learn well I wish you all the best and thank you.