Digital Systems Design with PLDs and FPGAs

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

Lecture-23
FSM issues 1

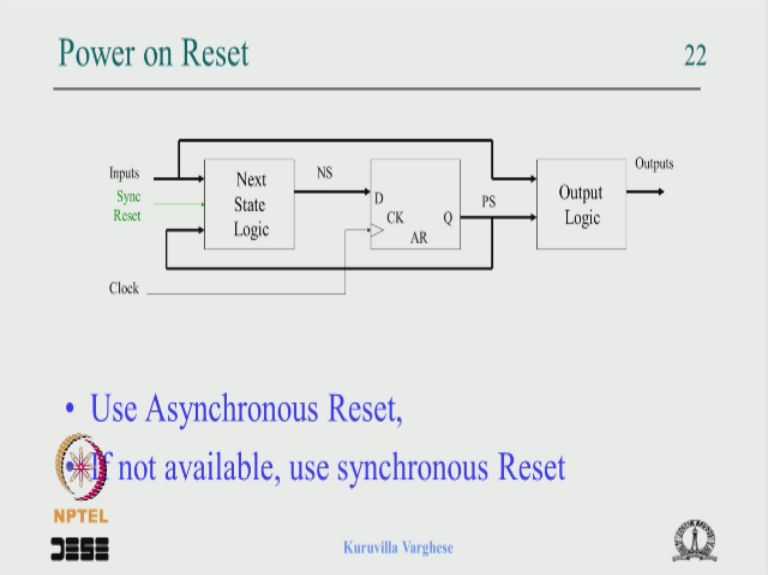
So, welcome to this lecture the course digital system designed with PLDs and FPGAs the last lecture we have completed the examples on VHDL. Then we started which some issues and the finite state machine are the controller. We have looked at how to bring the state machine to a starting state using a reset very simple thing. But if you forget then it can be problematic okay. This is like you may cable and you want to grim a connecter.

And normally there is an outer ring and if you are not put and you have gone ahead and grimt, then you realise that you are not put the outer connect a outer turn or something like that. Then it is tough so, something like that. So, you should not forget to bring the state machine it was starting state by you know putting the reset, second thing we have looked at is what is the kind of most appropriate clock frequency for a controller for a state machine.

We have looked at the maximum frequency but then that is not a good idea to run the state machine has the maximum frequency it dissipate lot of power, suppose the data path is running at a much lower frequency. So, what should be the kind of how much you can bring down the clock frequency of the state machine and what are the issues associated with it then that is what we have seen I mean we having completed.

But then we kind of set the case for minimum clock frequency, and we have seen some effect. So, I will just have a quick plans at gives last lecture slide. I will not go back to VHDL, because you are we have done a quite a few lectures on that, you must be through with it. So, let us move to the slide. We will see the state machine part as a revision, then we will continue. So, let us look at the slide.

(Refer Slide Time: 02:37)



So, this is the slide on power on reset. So, this is the state machine you know kind of structure. You have the flip-flops which gives the present state, and that combine with the input will, you know give the next state using next state logic. The present state is decoded to produce the outputs, some time, some output maybe just a function of present state, and some output could be a function of the present state and input.

And as we continue this lecture, we will have a look at what I mean what is the difference or what are the features of this Moore kind of output and mealy, kind of output we will see that. So, I said suppose the flip-flop you are using in the technology suppose you are implementing the state machine and FPGA. And the FPGA flip-flop as an asynchronous reset or you have making it an VLSI chip and the technology of flip-flop as asynchronous reset.

Then it is easy to use that asynchronous reset, it is very quick as soon as you assert the reset pulse with the minimum delay or are the present state become the 0. So, naturally the starting state can be all 0s, you know that is the most easiest thing to do. Suppose, you start with a state 001 then you may have to reset some flip-flops, set some flip-flops. So, when it comes to the state assignment, it is always easy if you can assign.

The starting state as all 0s okay depending on the number of flip-flops. So, asynchronous reset can be use. And suppose we do not have that asynchronous reset in the flip-flop. Then we have to

look for synchronous reset, the advantage is that the synchronous reset can be an input h to the next state logic, and the game is that if it is asserted. Then we may we say that the next state is 00 or the starting state okay h.

In this case you have an advantage that you need not reset to 0, when you have a sync reset since it is part of the next state logic you could make it easily to any state okay. But if it is 0 it is all the Moore convenient and we have also said in tune with a kind of synthesis we are doing at the RPL level. It essentially means like given to a synthesis tool and you know that the reset as priority over all other inputs.

Then it means that ultimately the circuit is nothing, but you have a 2 to 1 multiplexer at this stage with the reset as the select line. When it is 1 the 0 goes, when it is 0 the normal thing happens okay. That is I mean we have seen that kind of synthesis, and that is what even if you write the equation, and equivalent circuit is going to come at the output. Instead of a 2 to 1 mugs maybe a simple AND gate comes, and this sync reset can be, if it is active i, then it can be inverted.

And can be given to the input of the AND gate. Suppose if the number of flip-flops are 2, then there will be 2 AND gates. And the next state inputs are going to the d1, and d0 will go to one of the inputs of each AND gate, the other input goes to the inverted sync reset for active high and so on okay. I suppose that is simple enough for you to work out, I just give some detail about, how it is going to be implemented, when it is really you know worked out the equation of worked out.

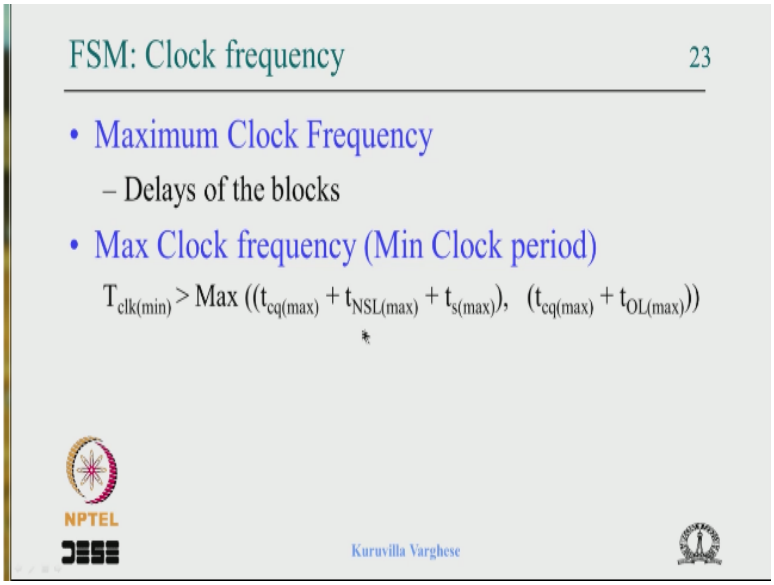
So, that is about the power on reset. And we have looked at the clock frequency, and the question was that, what is the ideal clock frequency for the state machine okay. Now, we know that what is the maximum, okay. It depends on this path and this path okay. So, this path going all the way up to of through some register or up to after here. And since, we do not where it is going. We will consider up to here okay. So it means that the clock period should be greater than the TCQ.

Because a clock comes it takes TCQ delay, then the next state logic delay, and that has to come here setup time before the next clock edge. So, the tclokmin, tclock mean should be d greater than tcq+t next state logic+tsetup or it could be tcq+toutput logic. Because even if it is it is

something directly use here say like it is it is crazy if this path works properly. But the output is does not get a chance to become you know active.

You know it before that it goes to the next state and I mean everything works internally fine. But the output kind of does not have time to get decoded, then it is a bad thing. So, that is the maximum frequency.

(Refer Slide Time: 07:56)



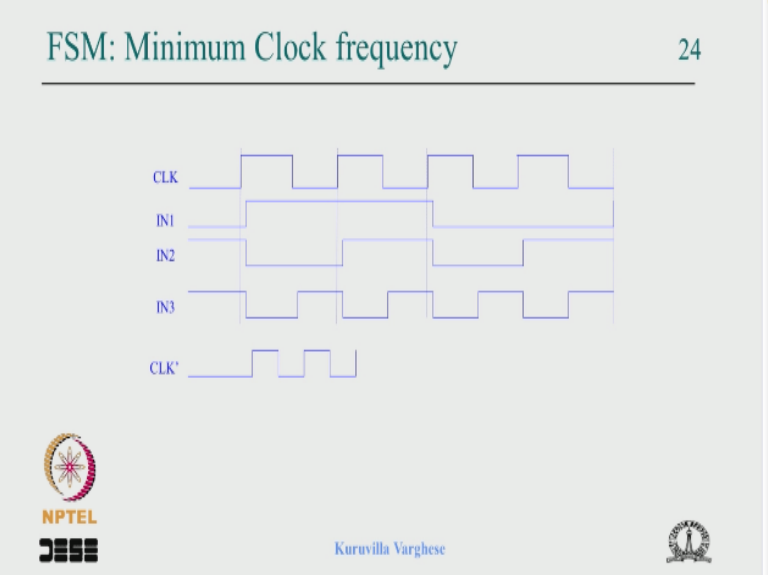
That is what is we have putting it here it is so, we choose max of two paths you know tcq max+tNSL+ts everything max okay and this is the tcq+tOLmax TCQ+TOL max and you should know that like a event when we use a tool. There are the clock period we use a maximum time delay and for the hold old time violation. We use the minimum delays okay, because it is the hold time violation happens with the same edge.

So, in the tool when you simulate you have the opportunity to choose the fastest delay, or the slowest delay and all that corner cases can be chosen, when you do timing simulation maybe, when we work with the tool. I will show you how to do that. So that is the maximum clock frequency. But then we set there is no kind of point in clocking at the maximum clock frequency, because it dissipates lot of power and more over.

We have some data path which is being control by this just controller. All these inputs are coming from somewhere in the data path and these outputs are going to the data path. So naturally and that data path is working at some frequency. We optimise or we maximise the clock frequency in the data path to get the maybe the highest through put in in some application okay. So, there is no point to kind of like suppose, that is working at a 100 mega hertz maybe it is few tile.

And you know lot of power wastage, if you work at 1 gigahertz clock frequency okay. So, the question is that, what is the minimum we can do okay like given, and known like if it is if the data path is given, then what is the minimum clock frequency we can choose, and that was a question and to illustrate, we have put I have put some kind of simple picture there.

(Refer Slide Time: 10:11)



Assume the data path the IN data path outputs which is going to the input of the state machine, that means again in this picture we have suppose 3 inputs IN1, IN2, IN3 coming from some part of the data path okay. So, I have put that okay, so and to understand the problem we had made it very kind of simple and very symmetric okay. We have put square wave like clock which is not the real life at all that would, but we can.

As we can as I said when you analyse always go for the simple case always go for the symmetric case, regular case. For that once you grasp the underlying issue, the basic how to solve it, we can

extend it extrapolate extrapolated it to the real life okay. I have sighted an example suppose you have working with the linear algebra with matrices and met results. And you have probably working with Eigen value values with either value or linear transformation or rank of matrix.

And if you work with the 5 by 5 metrics and try to get some intuitive entity feeling about it, it it may not be possible. But then if you say work with a 2 by 2 a matrix or 3/3 matrix, it is possible that you can come to a 2 dimensional or 3 dimensional graph you can draw with a vectors and you can make out suppose we are applying some linear transformation what is the happening to the vector can be easily understood.

Then if you if you grasp the basis then you go to 10/10 matrix that at least that clarity will come into the picture, you cannot draw a kind of a graphical picture of the 10 dimension. But if the matter is clear in 3 dimension okay, I am talking about the geometry, geometry to make the life easy, but need not be okay not our subject directly not a related.4

But then you know some people are comfortable to work with the algebra alone that you do not use such kind of isomorphism of going from algebra to geometry and back. But anyway geometry probably a limited with the 3 dimension and so on. But anyway coming back to our problem we put very simple symmetric case, let us assume here, I have chosen very cleverly kind of manipulated case.

IN1 is a some kind of clock kind of waveform, IN2 is half of that, IN3 is half of that the clock period. So, the frequency is twice, and this frequency is twice of that. So, this is the scenario this is going to the state machine and we have put something like this okay. Let us assume the state machine clock is this, and we have found absolutely no problem like at this positive edge the state machine is in some state.

It detected that IN1 is 0, then IN1 is change stain in the next clock edge yes it has transited may be we are in a state looking for this, input 2 go from low to high absolutely no problem. Because we say in a state IN1 bar you remind there IN1 you go the next state and that will happen

perfectly. Because at this stage it will be in that particular state, but when the next clock edge comes it fines that the IN1 is 1.

And will transit to the next state may be we will make some output to respond to this event, and that is how we control and here again it is 1, then it is 0. So, it is able to track all the changes in IN1 and look at IN2 absolutely no issue, when this clock edge comes IN2 is 1, then this comes IN2 is 0. Then IN2 is 1 no problem, but look at IN3 and what happens, you know the clock edge comes to the state machine.

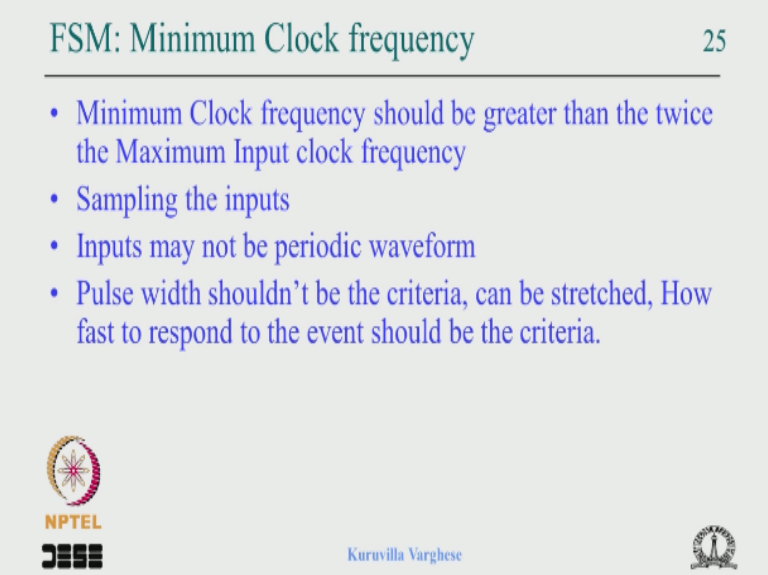
It detects a 1 suppose there was a, you know condition that as long as IN3 is 1 remaining 1 this state. And if IN3 is 0 go to the next state, and you see that it has gone to 0, then it has gone to 1. But at the next edge is still 1, in between it has gone to 0, it has come to 1, next clock edge our state machine still is detecting that IN3 is 1. And you get struck at that state, it will not even proceed to the next state, and it is struck there.

And waiting for ever for this particular input to go low. So you get the problem now so, this clock is not able to detect to change in this particular input for it to be detected. We know that at least one active clock edge should come in this period okay. Suppose an active edge as come here, we should have an active clock edge somewhere here. So, like you should have an active clock edge in this period and another in this period.

So, naturally one clock period suppose it is good if we can have one edge here and one edge here and so on. So, we should have at least one clock edge in this period that can happen, if the clock period of the state machine is at least less than this particular width. So, you see this is the maximum frequency input to the state machine. The clock period is from hereto here okay. So you take the half of the clock period.

So, the period of the real FSM clock should be less than half the period that means the frequency of the state machine twice should be that of maximum frequency input okay that is the game so that is it okay.

(Refer Slide Time: 16:31)

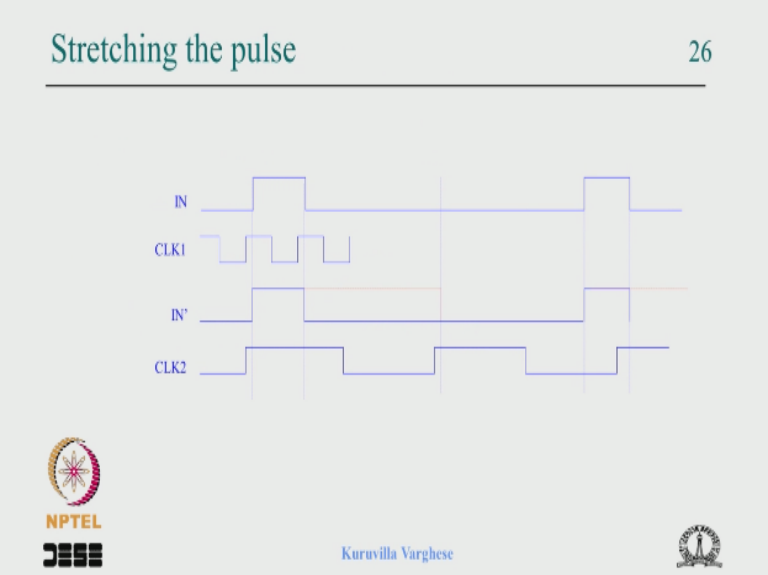


Now maximum clock frequency should be greater than, that twice the maximum input clock frequency okay, it is sounds like you know in a kind of (()) (16:44) criteria you say which definitely we are talking about the transform the frequency domain via way signal and you want to sample it. And you know the highest the clock frequency highest frequency contain the sine wave. Then you should sample at twice that.

So, that you can recover it something similar to that happens which should not be surprising, because we are sampling the input though we are talking about the digital domain where we are talking about binary signals not the analog signals and so on yeah this happens because we are sampling the inputs. And first thing is that now let us relax our criteria earlier we said okay.

We assume that it is fine it is a like a regular square wave, but in real life nothing like that happens so, it is sampling the inputs, input may not be very periodic waveform but you have a pulse width like you have an narrow pulse, that should not be the criteria for choosing the clock frequency. Because it can be stretched and the game is that you know.

(Refer Slide Time: 18:05)



I will show a picture suppose this is a input, say the pulse width is 10 nanosecond assume that it is 10 nanosecond so, you will have, you know suppose this is the next pulse come with after 1 microsecond okay. So, the clock frequency, the frequency of the signal may not be regular wave form but assume the minimum time period between them is a kind of 1 microsecond. Then we can say the clock frequency the maximum clock frequency is kind of 1 megahertz.

But this is 10 nanosecond okay. Now if you choose a clock period to kind of detect the change in this input, because we are this is coming to the state machine. We should detect that you know it has change the state okay. Now to detect that we need a clock period, which is only less than 10 nanosecond, that means this clock frequency will be 100 megahertz. But the input frequency itself is only 1 megahertz okay.

So, but the game is that you know we can kind of stretch this pulse because the next pulse come only after 1 microsecond maybe even possible to stretch all the way to 500 kind of nanosecond are anywhere okay. And you can use a much lower clock frequency to detect this change at the state machine. So, this is possible, but like depends on the application suppose there is a requirement some event has happen.

And this pulse has come but say the controller should respond to that event within say 15 nanosecond. Then it is you cannot arbitrary stretch it. We have to use a clock frequency which is

a kind of the period should be less than 15 nanosecond, because like after the input changes within 15 nanosecond the state machine should respond to it, then the clock frequency should be chosen according to that requirement.

So, that is what is written here, the pulse width should not be the criteria, it can be stretched how fast to respond to the event should be the criteria okay. So, all these the problem is that many a times you read in the text book, you get a very simplistic kind of solution this regarding all these issues. And even in real life, when you interact when you work with a specification many a times these kind of requirements are not clearly spelled even by your customer or the user are may not be clear.

You go head implement something then you deployed then you realise that so, it is very important to ask to be aware of these issues, ask the right question when you kind of form the specification. And it is very important when you design some complex system to write these specification the requirement in as much as detail as possible. And there are standard format requirement specification there could be standard formats.

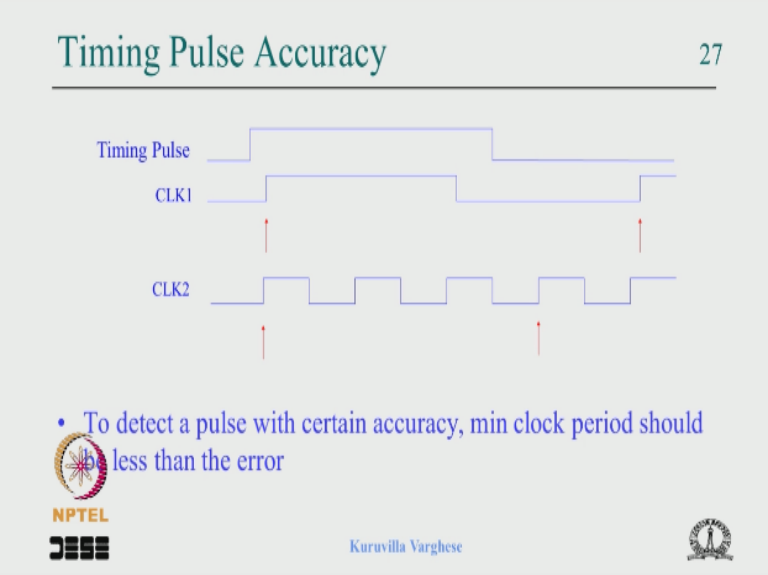
But I am not talking about the format you know these formats the standards and all helps, but even if it is not very organise, not very systematic writing it down writing the requirements clearly drawing some waveforms writing some tables bring clarity to the scenario of course you can use some kind of documentation standard. Because it helps to use standard tool to manipulate that it like in a team now a days it is all connected environment.

So, use a tool everybody can look at the tool multiple fellows work on the same kind of file and so on. So that is helpful. But the primary thing is to be aware of these issues and bring clarity to the scenario. So, let us look at the scenario particular scenario where there is a pulse, and I will show you a kind of circuit. How to do this stretching okay which is frankly not a very practical in the sense

There are data better ways to do it. So, let us at the beginning just for the arguments say see a simpler very simple circuit to stretch this pulse by whatever width you required as for your

requirement but I mean practically we do not gives that. I will show you a better circuit after that, but this is to stress the point okay.

(Refer Slide Time: 23:15)



So, one other thing is that here, we talked about a pulse being detected okay maybe and there is no timing great timing requirement. It has to be detected before the next pulse, then you can stretch it to any limit and you will detect it whenever you want before the next pulse okay. But otherwise you say the pulse width is 10 nanosecond, but it has to be detected within 500 nanosecond no issue.

You stretch it half way then you can detect it okay, but it may happened that, you may have to kind of a you have a timing pulse okay. And you have to detect it with certain accuracy. So, if you remember the case study we are you know discuss with respect to the ADC controller we had just before this lecture we had a case study where, we wanted to have a controller which controls an ADC to give the start of conversion to ADC.

And store the sample in a memory okay, there we said that the right to the FIFO should be a certain width and we have decided to put a counter outside and the counter will give a **a** count from 0 to some particular count. And we will decode that count and give back to the state machine okay, and state machine will detect that change. When it reaches the count, and stop that right pulse, you know that was a idea. So, we are in a similar situation.

I am showing in the picture of course I am showing a kind of again and **i** little ideal case. So, let us assume this is the timing pulse we have started this pulse at this point. And the state machine is looking for the end of the pulse okay. But say this is some few say 500 nanosecond but this being a timing pulse, there has to be some accuracy on it. Suppose, we are giving a delay like around 500 nanosecond.

And suppose the state machine is sampling this pulse with the clock period, because this pulse is going high here, going low here. Suppose we use a clock like this looks perfectly fine. Because this clock earlier would have detected that it is you know 0. Then this clock edge it is detecting that it is going to 1, then at the next clock edge it is detecting that is gone to 0. But our aim was to sum out capture this delay with certain accuracy.

But what happens now what we capture is that we will check at this active edge that it is 1. This active edge it is 0 so, what time we detect the state machine detect this time period not this time period okay. So, that is the issue, and suppose you use a higher clock frequency higher clock frequency. There are positive edges here so, we are sampling at all edges. So, here before it was detected it is 0 at this point it detects there is going high.

And you see here it is 1, here it is 0 so, here if you see that real timing pulse was this, and here the pulse width is this. So, we are much closer here, suppose the requirement is that suppose this is kind of assume that it is 100 nanosecond. And this has to be detected with a kind of accuracy of say 90% okay. So, the error should be 10 nanosecond okay so, assume that means this should be detected with 10 nanosecond.

So, it tells that the clock period should be kind of in that order, you know the clock period clock edges should come in kind of within 10 nanosecond. So, that is what is I have written to detect a pulse width with certain accuracy minimum clock period should be less than the error requirement okay. This is quite but when you do many complex thing you forgot you can forget this it is very natural.

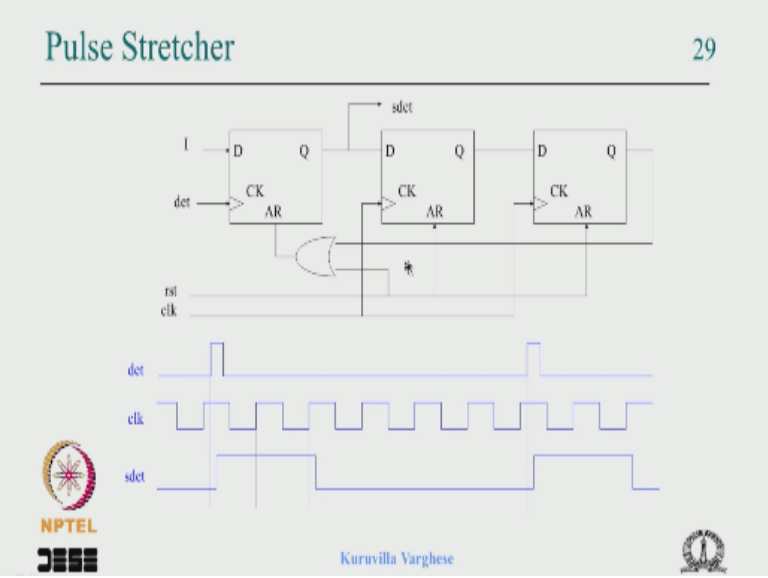
That if the counter is working with the 100 megahertz. And the state machine which is controlling the counter is working with the 1 megahertz, you know that there is something wrong, you know the counter is staining state at 10 nanosecond, and the state machine is looking change looking at sampling at 1 microsecond is something not acceptable. So, that is what is the game is so, this should be kept in mind as for as when you have a timing requirement.

But in any case but to make a assumption that a everything should be greater than any frequency used in the data path will be a over simplification. And nowadays you have lot of constraints and power the timing and all that. So, it is better to choose the optimum clock frequency which minimise the power dissipation and many other things like that. So, that is about the accuracy of a timing pulse.

So, let us look at this scenario we have a pulse and which is actually the frequency is low. But the pulse width is very narrow and if you just consider the pulse width you will end up choosing a clock frequency for the state machine which is very high. But if you stretch it, because the frequency is very low, then you are able to use a higher clock frequency sorry lower frequency for the state machine so, illustrate the principle.

I will show a pulse stretching circuit then we will come to more practical circuit okay.

(Refer Slide Time: 29:35)



So, let us keep look at this pulse stretching so, what we do is that this is a pulse catching flip-flop, you see this flip-flops and this is a pulse. In this case a narrow pulse come this is a input and with the very low frequency the pulse come again okay. And if you kind of concentrate on this pulse width, then you will choose a clock period which is less than this width and there could be very high okay.

So, assume that we want to stretch it to kind of somewhere here so, what we do is that we have a clock and reset but the clock goes to this flip-flops. But this catching flip-flop has the clock as this you know it takes this pulse as the clock. And d input is given to 1 okay so, what happens is that when the pulse come this s detect. So, I am calling this bit and this is synchronous detect, synchronous detect you see that it goes high okay.

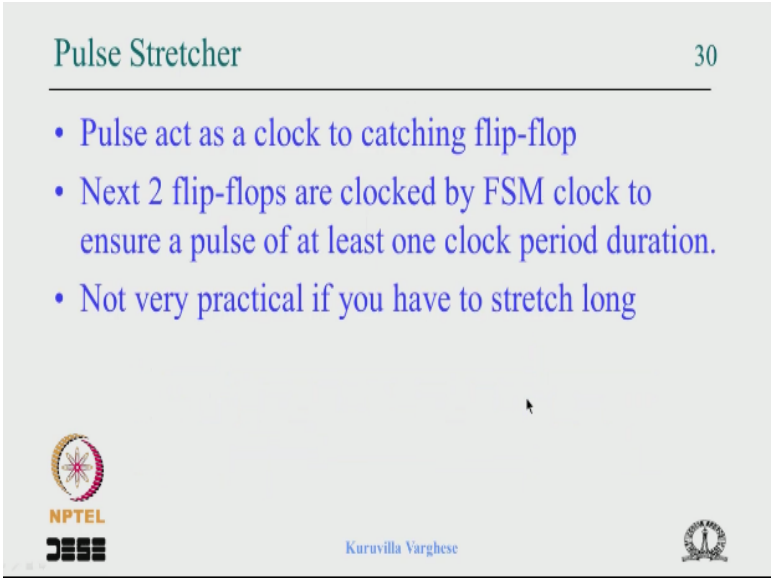
So, the detect comes on the next clock edge yeah so, here it goes high and the next clock edge that is shifted here. And in the next clock edge it is shifted here, that is what is shown here. It goes output it goes high the first clock edge comes here, the second clock edge comes here. The moment it comes here if goes back and reset this game, and this low okay. So this pulse is stretched by 1 clock period because it looks as if it is 2 clock period.

It cannot be 2 clock period, because we are not sure when the pulse, pulse is asynchronous to this clock. So, it can come very close to this kind of this clock can come very close to it, and that is why mentioned it is not a very practical circuit I will show a practical circuit. So, but this is to kind of understand the principle so, we give 2 flip-flops to get a 1 clock period stretch. So, that is how it is stretch at beginning it is reset everything is reset.

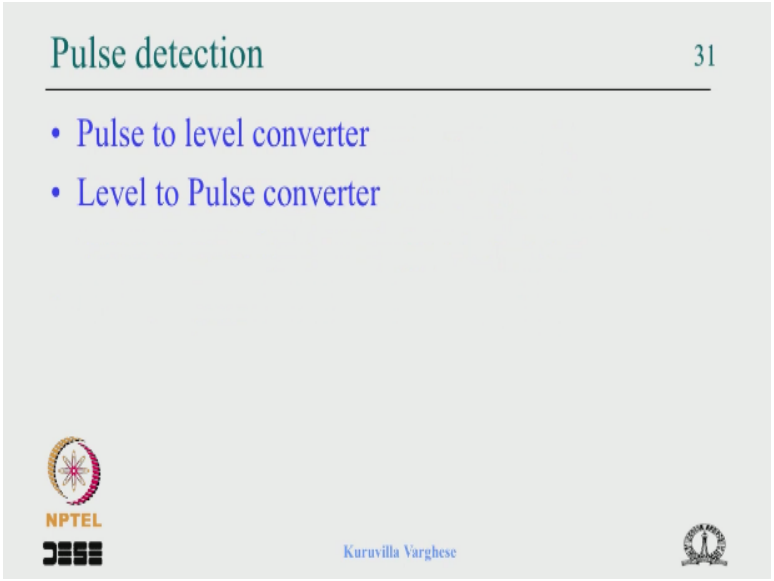
And then the pulse come it is kind of stretch at least by 1 clock period, you want to stretch it by 2 clock period you put 1 more flip-flop and so on okay, not a very practical circuit. Because there could be a timing issue at this point when the input comes here with this flip-flop maybe if you put a chain of flip-flops get resolve. We are not seeing what is the timing issue but so, I am not able to kind of discuss that now.

And it is problematic it is not a very practical circuit suppose you want to stretch it to some kind of 10 times of clock period. Then it is not a very kind of practical circuit. But then you might ask I mean why to stretch it 10 you know you reduce a clock frequency and so on. So, we will look at a more practical circuit.

(Refer Slide Time: 32:45)



(Refer Slide Time: 32:48)



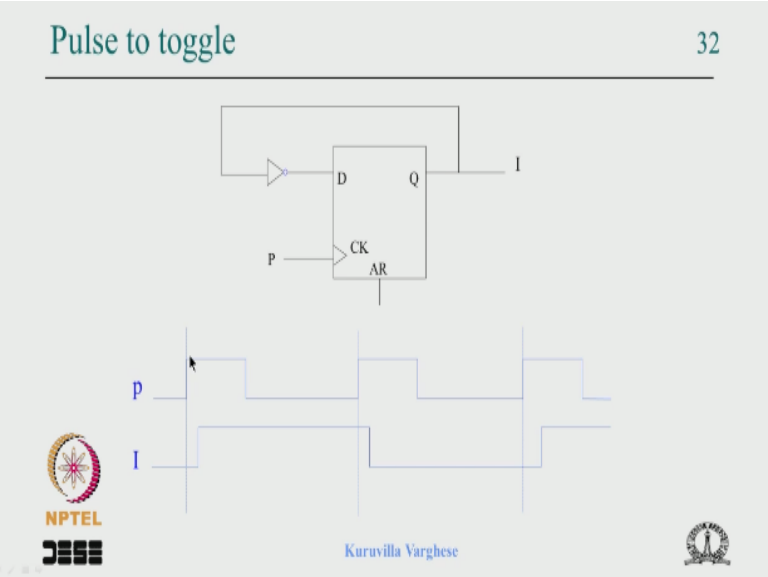
So, that is using something called a pulse to level converter and level to pulse converter. That means the basic idea is that when we have a pulse instead of just stretching it, when the first pulse come, we will make an output of this some circuit to go high like that continuously high.

The next pulse come that high is made low okay so, there will be between the pulses you will get a kind of square wave and what we do is that at the state machine.

When it goes high we will convert it into a pulse in terms of the clock frequency of the state machine okay which could be decided by some other input okay. We are not sure this input is very low frequency may be there are some other input which is higher frequency than this.

That will decide the clock frequency of the state machine and using that, this pulse going high that will be converted in the FSM to another pulse which is wide enough to be detected by the state machine okay. That is the game so, what we will see first how to convert a pulse to a level.

(Refer Slide Time: 34:01)



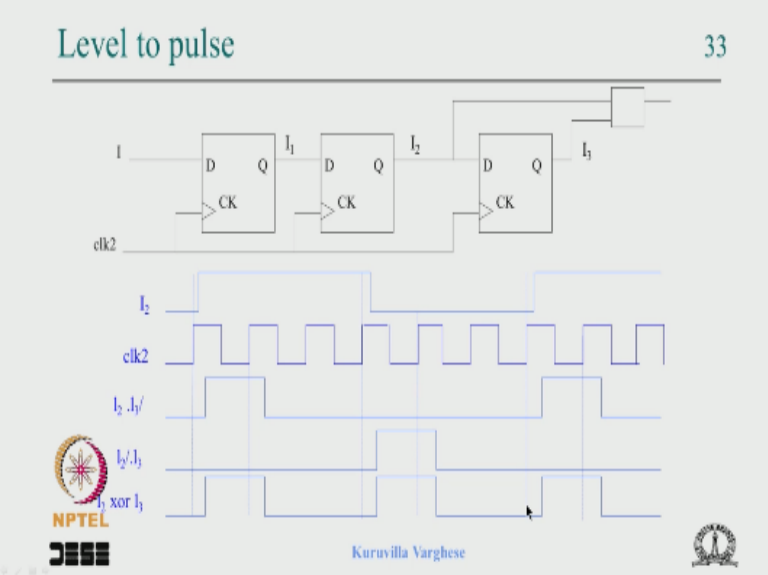
So, that is what is very simple. So, assume that this is a pulse okay, I am not shown lot of kind of gap but assume there is gap. Then the next pulse come next pulse come, and this is a d flip-flop. You see the pulse is given as a clock okay, and at the beginning it is reset so, you have I am calling this i because it is going as a input to the state machine.

That conversion logic so, I is low but when the pulse positive edge comes. You see this, this is a inverted and given so, that this is 0 at the beginning, then when this pulse come it becomes 1. And it remain there as long as the next pulse come, the next pulse come it was already output

was 1. So that is inverted and the 0 comes here and goes here. So, between the pulses either it is 1 or 0 now we take it as a state machine.

And state machine has some clock it is enough, if we get a pulse of that state machine clock frequency at this starting and this point or if it is negative edge you can make it here it does not matter okay.

(Refer Slide Time: 35:19)



So, I will show that kind of circuit and so, this is the input which is a level you know we have converter to pulse to a level signal or a pulse to a toggle signal. That is I and this is I that we are giving it here okay, now this is the practical circuit I am putting 2 synchronising flip-flops. We have not studied why the synchronising flip-flops is required that is to basically to meet the set up time here okay.

Otherwise the this flip-flop can get into something called meta stable team we have not added may be towards the end of the lecture end of the course. I will touch up on it, I hope I will have the time to get into it. But this avoids that probably it is not that this scenario is completely removed but with high probability that is kind of that scenario is averted that is the basic game so, do not worry about this at all.

So, assume that i comes here it goes through a double state synchroniser it does not matter as for as timing is concern, i1 is a delayed version of I, I2 is a delayed version of I, you know I1, so, I2 is kind of 2 clock period delayed version of the i okay that is the game. So, we will see so, the pulse come you now like this a toggle will comes so, assume at the starting point there was a pulse here and there was a pulse here.

We have converted to toggle now, this is where the all transformation happens, we have an input a flip-flop which input and output is combined through a logic and assume that the logic is that i2 and i3 bar okay. So, you assume an and gate here I2 goes straighter the and gate, I3 goes through a bubble or an inverter, I3 is inverter see what happens. So, assume that this is the clock to the state machine and we which we use in the same this transformation circuit.

And the see the i2 goes like this, okay here for analysis that is enough we do not have to analyse i and the clock comes. And when the I2 comes you see the clock comes so, the I3 will go high okay in the next clock edge. But you see that I3 we are looking for a scenario where I3 is low and I2 is high. So you see I2 is high here, but the next clock edge only the I3 will become high so, at this point with the 1 clock period.

There is a condition that I2 is high and I3 is low, because this clock edge only the I3 can go high, because it is at the output. So, i2, I3 will give a edge, a pulse which is of the duration of the clock period with the slight shift. And that can go to the state machine and the state machine is using this clock so, naturally it will sample and detect the pulse correctly okay. So, it is a very clever circuit.

I hope you got the picture this was I2, I2 was 0 before and in the next clock edge here only the I3 will be become 1, so before that. So, during this period I2 is 1 I3 is 0, so you get a pulse and that only happens for this 1 clock period, it does not happen here, it again happens here. Suppose you want to catch the if you want a pulse at the negative edge you do the opposite.

That this point is low and that point is high and that happens at the negative edge, you can kind of analyse it. and suppose you want the pulse at both edges. Here and here for some applications

you know that it is nothing but I2, I3 bar or I2 bar I3 which is nothing but I2 xor I3. So, depending on what you put here maybe an AND gate with a bubble here, or an AND gate with a bubble here, or an xor gate you get any of these pulses.

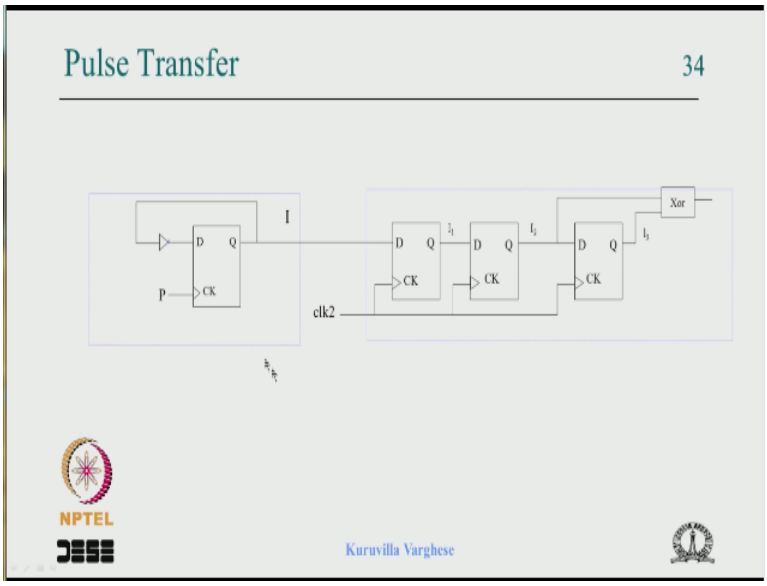
For our case we are going to use this okay. So, we where we have a pulse using this particular circuit you convert it into a level signal okay. And this input is goes to a double state synchroniser to avoid some timing issue with this flip-flop. And then in this flip-flop we combine the input and output through a kind of this logic very simple logic then you get the correct pulse and very important the timing is correct.

Because we are you know deriving with a delay with respect to this clock edge that pulse will come with a delay. And when this pulse ultimately goes to the state machine, state machine will correctly sample it as you know that is very. So, this generate the proper timing as for as the state machine is concern. Because state machines working with a the same clock that is assumption.

So, I hope you got this picture it is a very useful kind of structures which is very much use in synchronising clock domain crossing only difference is that, I have little bit manipulated this and when this pulse is coming from a domain with another clock. Then there could be a slight change in the circuit, the pulse can go here in the data path.

But the clock of the domain comes here you know it is a very slight change, it is a very useful kind of principle and put together. So, I am putting this at the input. And side and this as a output side and I am combining it.

(Refer Slide Time: 41:33)

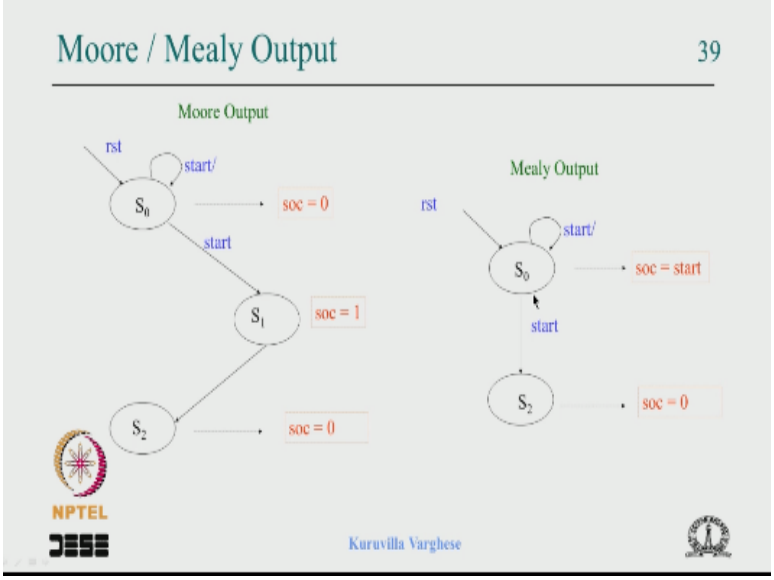


so, this is what which does trick or the magic you convert a pulse to a level signal and you synchronise it. And you convert level to pulse which is in this clock domain properly the timing is proper with this clock domain. So, that is how the pulses are handled and so, essentially we are saying that though we said the FSM clock frequency should be twice that of the maximum input clock frequency**.**

we should not be worried about the pulse width we should be worried about the frequencies and if the pulse width if there is pulse it can be stretched to be neatly detected by the state machine okay and we do not like the first circuit we have shown, it is kind of stretching it integral number of the clock cycle. But then this is a better scheme you have not even stretching it, we are toggling it okay it is a kind of extreme case of stretching it.

We are stretching all the way to the next pulse you know. So, it is like a extreme that a pulse is stretch from the first pulse is if the second pulse. Then it is kind of made low, it is made toggle and then at the other end we convert back into a pulse which is of correct which has a correct timing that is a most important thing.

(Refer Slide Time: 43:08)



Now let us look at this Moore and mealy output. So, there is a confusion where the Moore output should be use, where the mealy output should be use. If you read a textbook many a times it appears as if you are a making a big choice at the beginning of the design say because many a times it is called Moore machine and mealy machine. And that is very confusing you know it is as if at beginning of the design process you are assuming okay.

Let us go ahead and design a Moore machine it is not so, because there are in practical controller there is lot of input lot are outputs. Some outputs are function of the present state some are function of the present state and input. So, it is not a question of a kind of designers preference to choose Moore output and mealy output, that is what I want to bring forth.

There are places where the mealy output is kind of works properly and best option then is to choose mealy output okay. There are places where the mealy output cannot work okay. The accept places we should use the Moore output. And the discussion may not be complete at this point. There could be questions ask which with a present background I cannot answer your everything maybe as we go head in the lecture maybe some issues are handle, some issues are not handle.

But anyway we are improving our the grasp of the situation as we go along. So, let us look at our case study where we have control of our ADC. I will take a kind of simple case from there, there

the state machine was generating a start of conversion pulse okay, so that is the scenario. So, let us look at the state diagram so, this was the state diagram. So, let us go to the slide so, if you remember.

I hope it was simple enough so, in the, at the power on we come to a state okay, and we are waiting further start signal from the host CPU, as long as the start was low. We remind in the state and the SOC was the start of conversion pulse was 0, when the start signal came the state machine or the controller made a transition to the next state, state1. And is particular output I am only showing the kind of the concerned output or the relevant output okay.

So, SOC was made 1 and it is a unconditional transition the next clock pulse we go to a s2 and make SOC 0 and we were waiting for a end of conversion here okay. So, that is a clear picture and since this is an unconditional transition so, upon the start like when the start is detected on the clock. It comes to s1, the next clock it goes to s2. So, you know that the pulse width of this SOC is clock 1 period.

Because the minimum time a state machine can remain in a state is 1 clock period, because everything happens in terms of the clock period, in terms of the clock edge okay. So, this is the Moore output so, if you see that is the SOC is a Moore output which is a present state is decoded that means that when you write the output logic we say s0 state0. In our discussion we said it is 2 flip-flops the q1 is 0 q0 is 0.

So, we say in the output the present state is 00, output is 0, present state is s1 which is 01,the SOC is 1 present state is 10 SOC is 0 so, it is Moore output during the s1 the output is 1 okay. Now I am trying to convert this particular SOC into a mealy output okay so, how we can do that. So, what we do is that say because we have only one input the start signal, what we will do is that you look at the scenario, when the start is low.

It is machine is in this state SOC is 0, when the start is 1 transit to the next state and made make associate 1 in the state and transit to s2. So, we will do like this say we will say the machine is in a s0 state, as long as the start is low remain there. Now we say instead of SOC is 0, we say SOC

is 1, if the start is 1 okay so, SOC is written as a function of start. SOC is 1, if the start is 1, then we say when the start is 1.

We transit straight to the next state so, that is what I am showing here, the mealy output is upon the reset comes here as long as start is low remain there. And SOC is equal to start that means in this particular state, as long as start is low SOC is low. When the start comes high the SOC is high that means this SOC is a mealy output which is a function of the input start signal and the present state okay so, we are decoding the present state.

So, when you write the equation when you write the table we say s0 which is nothing but 00 and we have an input column where restart start is 0 then SOC is 0. When in 00 start is 1 SOC is 1so, in the equation of SOC you have q1 bar and q0 bar and start okay, that is the equation of SOC. And now we do not need s1 as the start comes the same state the SOC is goes high and it transit to the next state and SOC is made 0 okay.

So, that is the scenario that is how the mealy output is generated now we can see an all already you get a picture what can happen. And if you are a clever a clever student can already make out yes there is an advantage saying that this particular state is kind of knocked off. And you get 1 state is less and you can imagine if there are, this is we have talking only about 1 particular output.

If there are 10 outputs which was all kind of Moore kind of output, suppose we are a able to translate you know convert all that into mealy output maybe the 10 states are less. And that is the great saving and 10 states are less if it is a binary encoding lock 10 to the base to it can be, you know 3 flip-flops could be less. You know in the game depending on the start of state so, there is an advantage we clearly see number of states are less.

But this already should give you some hint as to what can go wrong like we are waiting in the state. And the start comes then the SOC is 1 so, that should that description itself should give you some picture of what can go wrong. So, we are coming to the end of the lecture so, we will see

that timing in the next class little more elaborately. So, because as I said we are seeing which is optimal scenario where we can without you know kind of any timing issue.

We can use mealy output and get some advantage and where we cannot use it we should be using the Moore output is what we are discussing. So, we have seen how to convert a Moore output case to a mealy output case, we will look at the timing issue in the next lecture. So, we have looked at basically the minimum clock frequency and the mealy and Moore output in this lecture. Please revise grasp the underlying issue I wish you all the best and thank you.