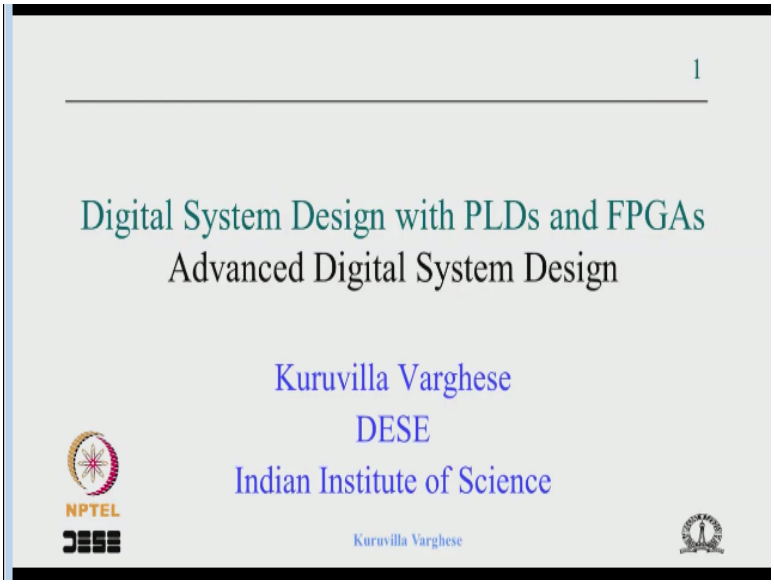
Digital Systems Design with PLDs and FPGAs
Kuruvilla Varghese
Department of Electronic Systems Engineering
Indian Institute of Science-Bangalore

Lecture-24
FSM issues 2

So, welcome to this lecture on advance digital design, digital system design in the course digital system design with PLDs and FPGAs. In the last lecture we have seen how to kind of decide the minimum frequency of a state machine and with regard to the data path.

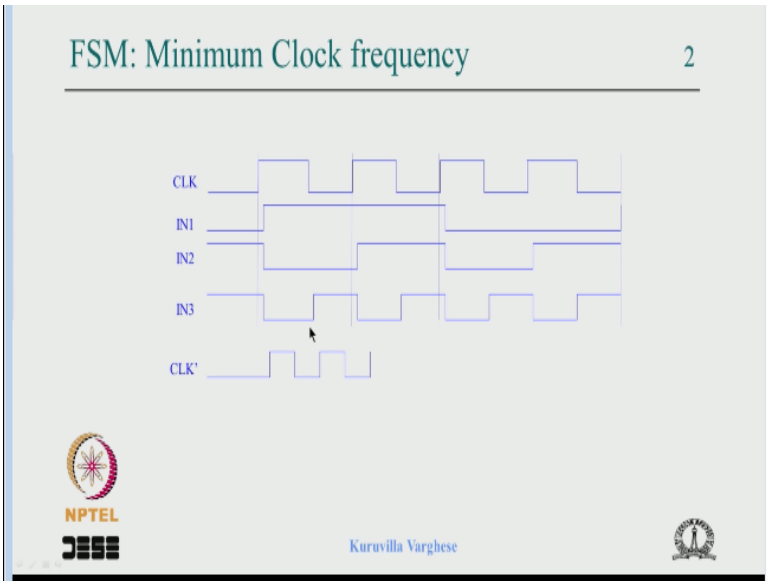
(Refer Slide Time: 00:47)



And also we have looked at the conditions you know the inputs are not very neat waveforms and when there are the real life inputs are there how to handle it because it may not meet some kind of requirements we put forth. And also have started looking at the features I mean what is the difference between Mealy and Moore kind of outputs.

And maybe how to at least in some cases how to make what is a Moore output into mealy output or at least to illustrate that is what we have done. We have a converted a Moore how to output to mealy output and I have seen what are the advantages what are the disadvantages we have not completed it. And we will complete it and today. So, let us look at the last lecture slides and we have you know looked at the minimum frequency`

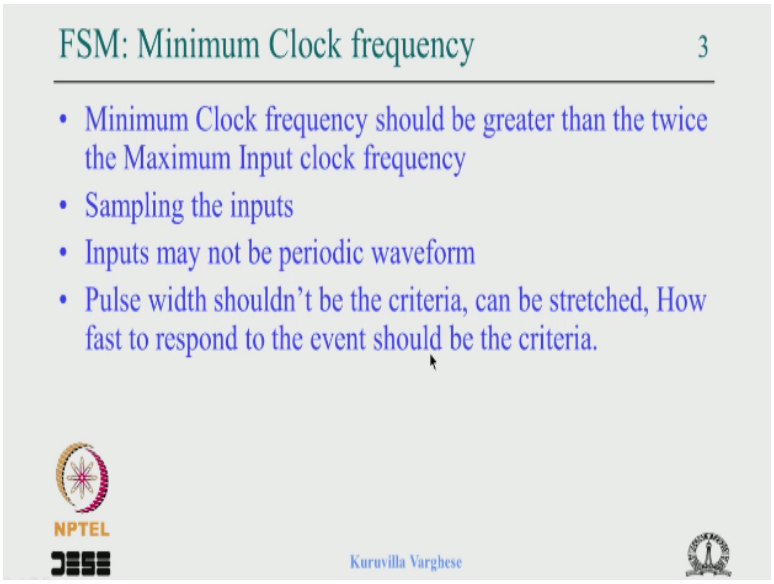
(Refer Slide Time: 01:52)



And we said the main game is suppose if you looked at 3 inputs and put this clock then we soon find that this clock is not able to detect change in this input 3. So, it is like we need at least 1 clock edge active clock edge during that change and in 3 and 2 kind of an ideal example we have taken some square wave. And it is soon clear that at least 1 clock period less should come within this kind of change.

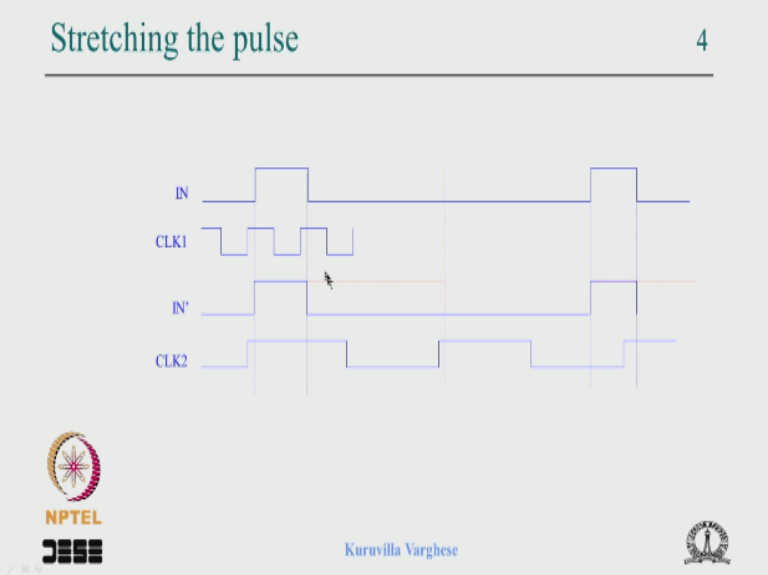
And change being symmetric or regular we can say that the clock frequency should be at least twice that of the maximum input frequency, that is a kind of thing we have told.

(Refer Slide Time: 02:43)



And that is not surprising because state machine is basically sampling the input looking at the inputs for any change any event, and trying to respond it with the output sequence of output sometime that is why it is called sequential circuit, and input in a real life, input is not a square wave of periodic waveform. But the pulse width should not be the criterion.

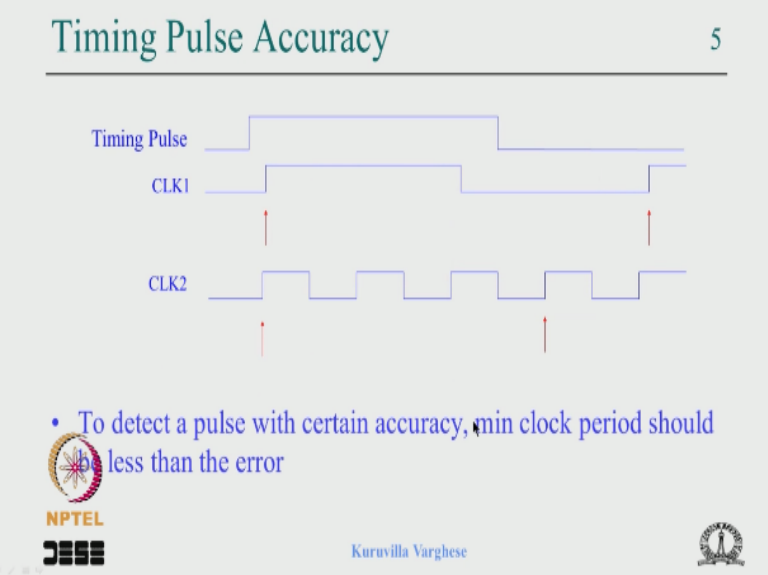
(Refer Slide Time: 03:14)



We that that means that you have an input which as a which changes for a while, you know it becomes 1 for a while say few nanosecond. Then say 1 micro second then it is inactive again it comes like for this stain to be detected in a kind of very straight forward way then one would expect to have a clock period kind of matching this pulse width but we know the we have we talked about the frequency and we know that the frequency is quite low.

So, definitely this pulse width is not the criteria for choosing the minimum clock frequency of the state machine or the controller also the kind of waveform suggest that there should be some way to stretch the waveform and then suppose in this case if you stretch it here, you could definitely use a lower clock frequency for the state machine.

(Refer Slide Time: 04:15)



And we have seen some practical example of I mean how to pull, how to stretch it, but before that I said something much more important here the requirement is only to detect the event, detect that it is input as change the state okay and respond to it. But maybe in some cases, you have a kind of accuracy requirement that means that there is a timing pulse and say we are trying to detect say it has gone high in respond to some event.

The state machine has made it high are enable something to something which is made this signal to high as waiting for to go level which meets some precise timing. And in that case if you use a clock like this it will detect that it is gone high here. And it will detect that it is gone low perfectly okay if just we are trying to recognise that it is gone high and gone low. But if you need some precise timing to be kind of measured.

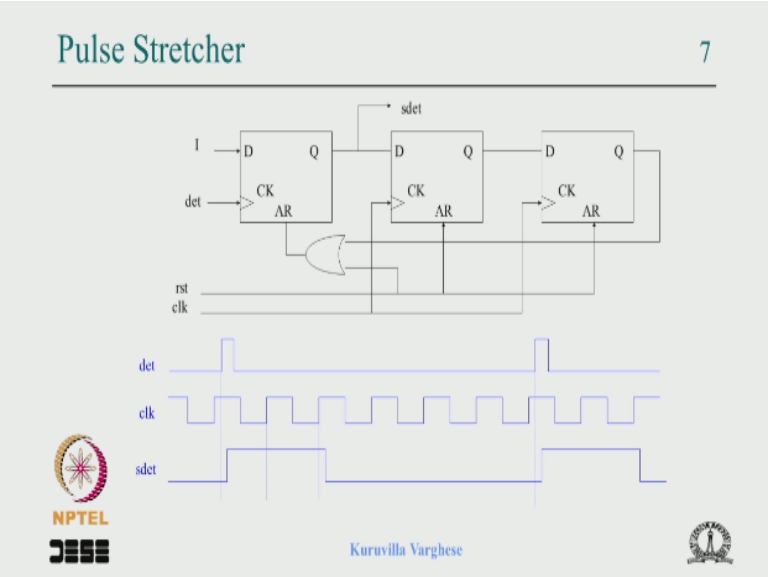
Then you know that this clock is much better, because this clock the detects that it is gone low after such a long time. But this much faster so, that means suppose if somebody is a there is a + or – error okay within which need to detected or within which an event has happened within which that has to be responded also, whatever then this is the event.

And it should be detected with an accuracy of some kind of time period then you know that the clock period should match that error okay. Then only we can detect it that is what I mean by this you know, and this situation arises, you know you have an event and somebody say controller

has to kind of respond to it within a certain time okay. It is nothing do with the pulse width maybe the pulse, itself remain high for a long time it is a very rare event.

But the requirement is that the event has happened, but this may next event may happen after 1 second. But somebody say that the response should be as fast as say kind of 1 nanosecond. Then definitely you need to detect to that accuracy need that clock period to be 1 nanosecond, otherwise there is no way to do that okay. So, that is what I mean and now let us come back to the kind of clock stretching.

(Refer Slide Time: 06:51)

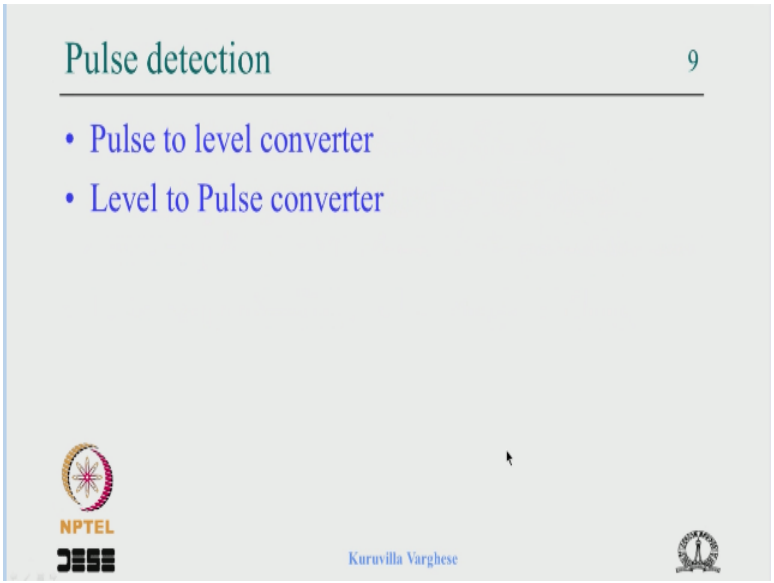


And we have seen 1 kind of circuit but which not very much use I have just shown something the illustrate. So, this is the pulse catching circuit where the clock is this is the kind of pulse. We are trying to extend it that is the clock and the input is 1 what we do is that we use a 2 stage shift register. So, that we are not sure, because this clock and this pulse is not synchronise so, we are not sure when this will happen, and that is double clock.

So, that at least you get one period width here the synchronise pulse this one so, in the picture it is shown as almost 2 clock period. But you know that this is gone up this edge could have been very near to it, and then you get only one clock period. And that is shifted and it is reset okay so, basically it is stretched and if you put a multiple stages of flip-flops. You can stretch it any longer, but then if depends on the clock of the FSM.

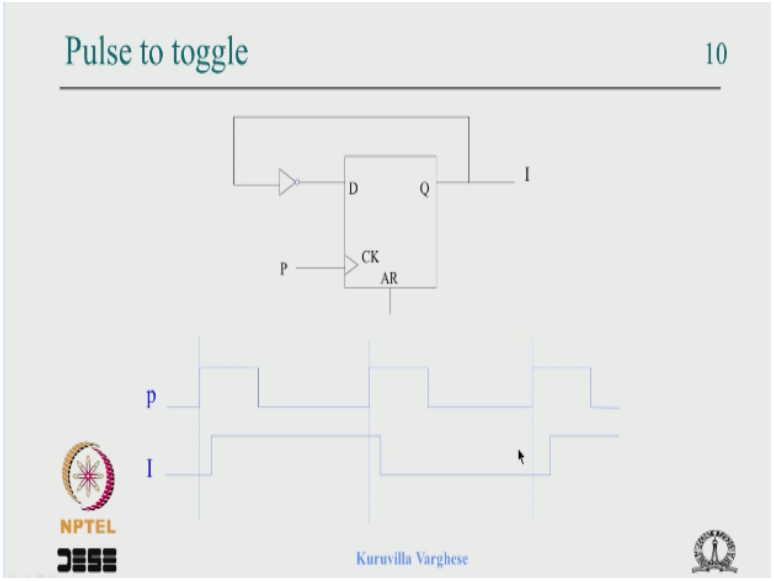
But if you use a lower clock there is no reason to kind of stretch it, you know more than 2 stages. Because we have one clock edge you know coming in that period, that is all what we required kind of to be detected by the state machine. So, there is practically there is no reason to put more flip-flops. Because after all, this clock is the clock of the state machine. So, 2 states should be enough

(Refer Slide Time: 08:25)



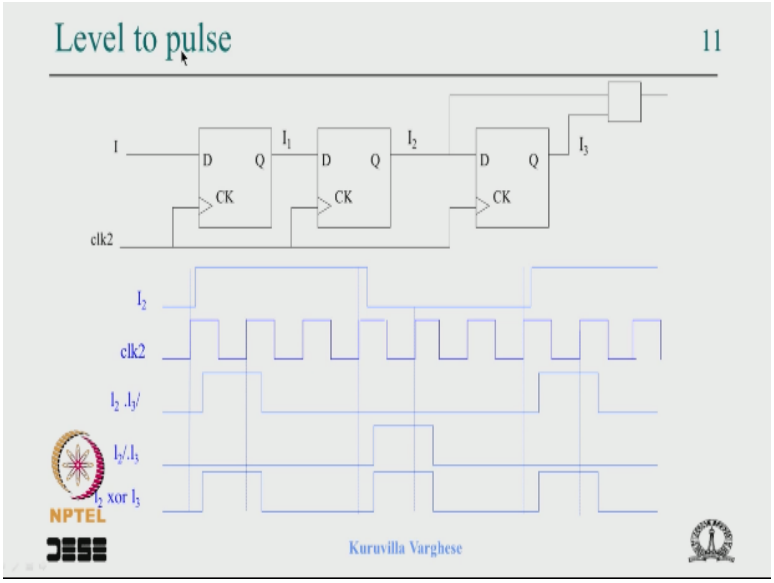
I have shown a practical circuit which is not stretching it what we do is that v as output when a bus come the output goes high and stay there till the next pulse come, when the next pulse come it goes low. And it goes to the state machine and what state machine does is that every edge every positive and negative edge you can make kind of a pulse.

(Refer Slide Time: 08:52)



And that is the first to level converter this is the circuit you have. The pulse clocking the flip-flop and you have a kind of toggle flip-flop the q is fed back to d through an inverter. So, first pulse come it is high next pulse come it is low and our aim is to get a pulse here at the receiving end and one pulse here okay.

(Refer Slide Time: 09:16)



So, we do a synchronisation for timing reason to avoid a metastability in flip-flop. We have not discussed it , but at least for the timing you just try to put a double stages synchroniser which is nothing but same clock and you two 2 flip-flops in series. And here is what we do you do and i2 the input of the flip-flop and output of the flip-flop, you combine and if you do i2, i3 bar, you get a pulse here, you the opposite i3 i2 bar.

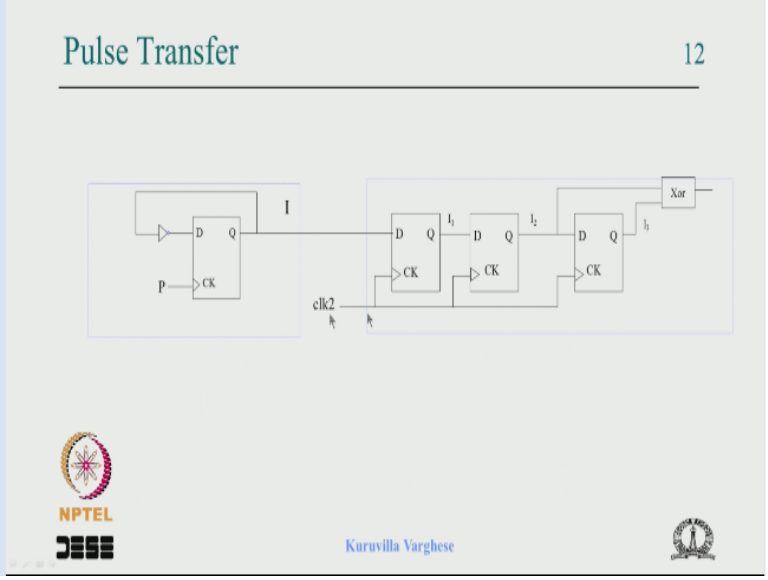
Then you get pulse and the negative edge and we want pulse at both edges. Because maybe in the last class I did not kind of stress it, because here there is one pulse, then it goes high next pulse come it goes low. So, at the receiving end, we need a pulse here, and a pulse there depending on the, which is of the width of the state machine clock. And that is what we have done this is the clock of the state machine and you do say you do i2, i3 bar.

You get a pulse and positive edge i2 bar i3 you get the negative edge you are it, then that becomes i2 xor i3. So, this is an xor gate then you get pulse here, pulse here you know that it comes with the delay of the clock period. And, that perfectly works fine for the state machine, because this is high during the active edge of the clock and that is this is correctly detected by the state machine. So, this is a very practical circuit very much use.

Though there are some kind of timing constraints probably which at this stage we will ignore it is not that you know it works for all cases you know there are some restriction on because of this double state synchroniser because you know that for the i2 to come here it takes 2 clock period okay. And so, that means between these 2 pulses there should be at least 2 clock period gap of the state machine okay not at not nothing do with the pulse itself okay.

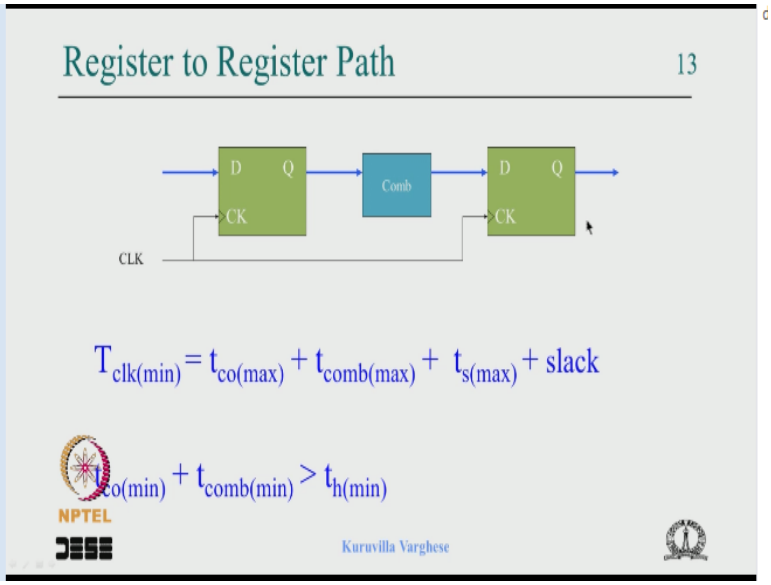
After the pulse comes here you know that it could be pass through 2 stages of flip-flop. And that introduce a latency maximum of 2 clock period. Because this is not kind this the input pulse is not synchronised to the clock. So, it may take maximum 2 clocks cycle, so it is important that the gap between these 2 pulses is at least 2 clock period of the state machine clock or the receiving domain clock we can say**.**

(Refer Slide Time: 12:07)



So, that is a circuit and that I have shown it you know in a combine together you have a kind of level to pulse to level circuit or pulse to toggle circuit and this is a toggle to kind of pulse back circuit.

(Refer Slide Time: 12:22)



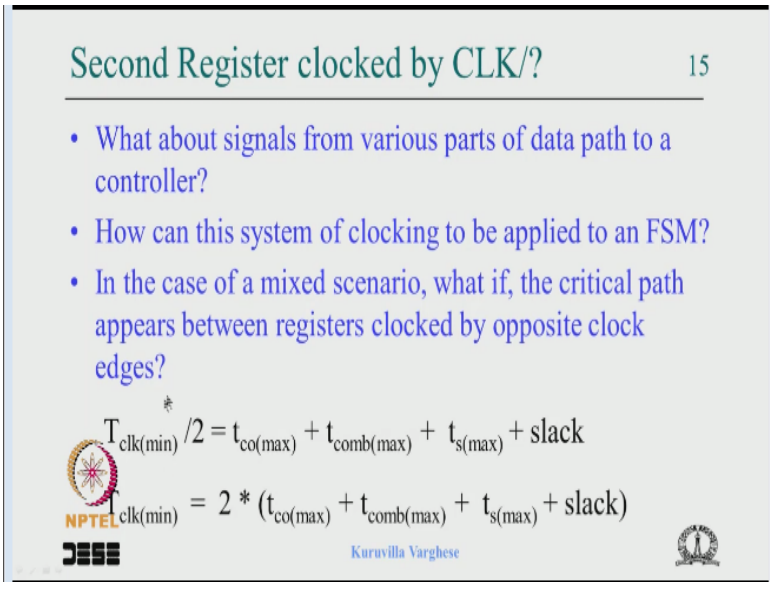
And we have discussed when we have a register to register path, b it is a state machine or a sequential circuit or a data path where a register a combinational circuit register in the case of the state machine or counter. This could be next state logic in the case of data path is some computation. But the clock period is chosen as tco, tcomb and tsetup all maximum. And tclock period should be greater than that.

And avoid hold time violation this delay + this delay which is minimum delay should be kind of greater than the minimum hold time. And very many a times people ask naive question why is cannot be in the clock bar as I said there is no great issue with the like the trouble is that you have looking at a part of the whole and if you have an old picture put in mind.

Then you will soon realise that this is not possible because this is clock, clock bar and again next is clock, that is fine but then there is a feedback from back here. Then you find this is the last one was clock and the feedback is also back to the clock not to the clock bar. Similarly you have a path here and it terminates that clock bar and you have another path here which terminates at clock.

Then it both output go to another register, then you have confuse where that should be either clock or clock bar. But cannot be both then you have stuck okay. Similarly like you have a data path giving the signal to the state machine and maybe state machine receive output in respond to a positive edge to get clock you know register and a negative edge registers. So, there is a big issue. So, **you can have** you have to stick to 1 priority it can be clock or clock bar.

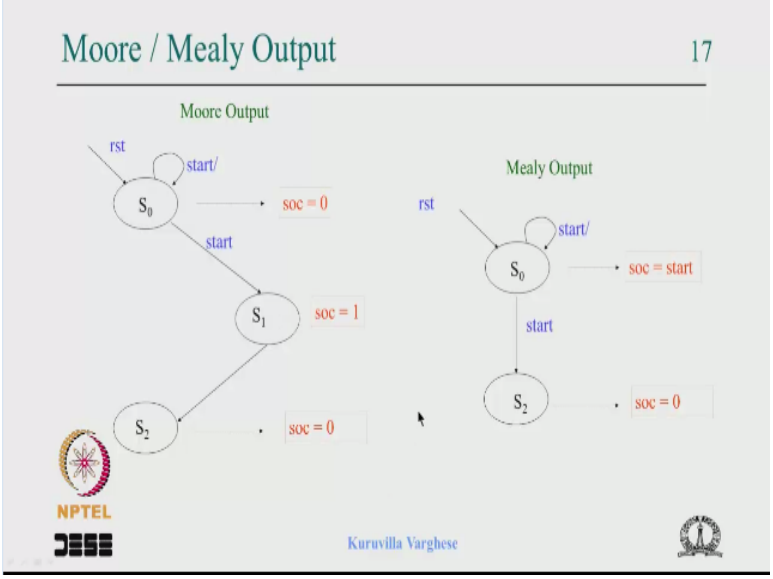
(Refer Slide Time: 14:26)



And timing wise and we have seen there is no big deal because if you use the in like. If it is possible to use half the clock period then this tclock meant 2 s to accommodate the maximum delay. The only thing is that the clock period will be you know twice that of the clocking by

single edge. So, the clock frequency comes down that is the only kind of advantage. But practically it is not possible to do that. That is a simple answer.

(Refer Slide Time: 14:56)



And this is where we have looked at and we were trying to kind of understand the difference between Moore and mealy output**.** Because in most textbook would deal with very simple cases of state machine with 1 input and 1 output and somehow many a times it appears that you know at the beginning you can kind of decide the state machine it is mealy or Moore.

And accordingly you at least that is the kind of it may not be intended by the textbook. But, that is the kind of picture many a times q does not get, but in practical cases there are as I said there are maybe so many states maybe more than 20 states more than 15 outputs and some are Moore some are mealy and one it is not a kind of ideological or some matter of taste to choose the mealy and Moore output.

It is not that you like mealy or something like that it is there are some cases where mealy is the most appropriate thing and some cases it cannot be done at all. So, you have to kind of choose and that is what we are trying to do this. And this is an example which is drawn from our case study that is why it takes the case study. So, that you can illustrate the various kind of concept based on that ones you understand that.

So, this is our adc kind of data acquisition controller. So, there when we discuss we said that at the power on it comes to the state 0 and the when the start it is waiting for the start for the host processor and when start is low remind there and we are only looking at this particular output which is of kind of concern. And the start of conversion process 0. And when the start comes it transit to next state.

And there the start of conversion is made 1 and there is no condition on that state next clock it transit to the next state and SOC is made 0. So, this state remains there for 1 clock period. So, you get a pulse of width 1 clock period and you make it 1 okay, now let us think whether we can make this as a this is definitely a Moore output. Because this is the decode of state 0 this is the decode of state1 and so on.

So, if the SOC is 1 we know that in the state diagram SOC is 1 only in 1 state. So, it is decoded as we know that this is 00, this is 01, so it is decoded as q1 bar and q0 okay that is all, that is a equation that is a decode of the present state. So, we had trying to do is that we know that already that this SOC as something do with the start okay when as long as the start is low SOC is low when the goes high the SOC becomes 1.

So, why not kind of (()) (18:18) Mealy output in that way okay. that means that like this you are the machine in is state 0 at the power on, it is waiting for the start signal. So, as long as start is not active remain in the state and we say instead of saying SOC 0, we say SOC is 1 if start is 1, in this state not we are not transiting to another state okay. So we are hoping that sometime being in this state the start becomes 1.

So, that state you know that state that means q1 bar and q0 bar that is 00. And start is the decoding of SOC, it is a mealy output and also upon the start it transit to next clock edge, it transit to the next state and are skipping s1, because that is done here. So, we skip to s2 where the SOC is 0. So, that is the mealy output, that is why it is converted. So, I think you get the picture and I said already you see an advantage there is 1 state less.

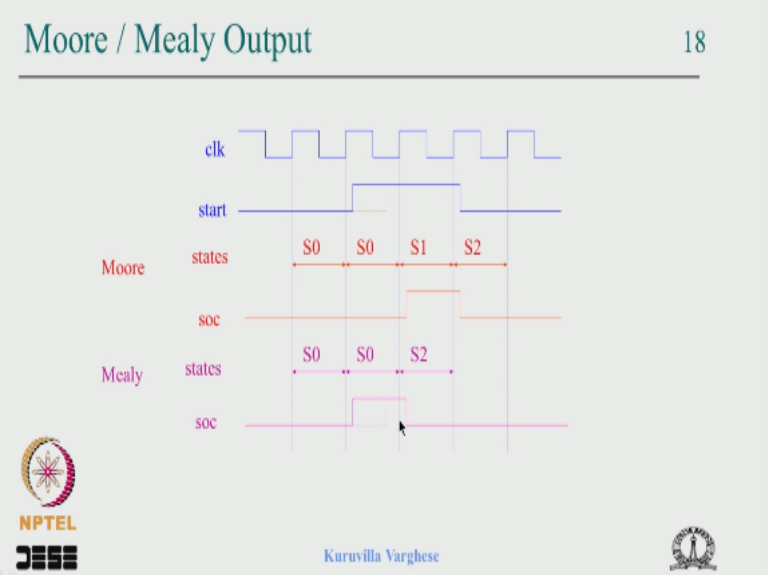
And you can imagine in a huge kind of state machine where there is lot of kind of outputs which can be made mealy output. Then you get states less, so number of flip-flops could go less and if the number of flip-flops are less maybe the next state logic. Because it is a function of the present state and input. The area of that could become less and the output logic which is a decode of the present state and input that becomes less and so on okay.

So, the area can become small the state machine can be clock very high you know high frequency it occupies less area reduces a power dispassion and so on. So, so many advantages one can think of and as I said that for a clever student like as advantage there already should some indication of a disadvantage you know a that is an timing issue, because you are remaining in this state.

And when the input goes high the output is high okay. That already should give you a kind of hint has to what can go wrong. So, let us kind of illustrate it much more than that this is not a quiz, so I **I** want that concept to be ingrain in your mind. So, that you design something I do not know what we are going to design maybe you design something trivial then it does not matter like if you design a computer game where there is a chip.

And if something misfired it is not a great deal. But if you are sending a rocket to the space and if a slight mistake could put things in a jeopardy. So, maybe a you have part of an aircraft which is being designed and there is a controller which you have designing as part of something guidance, then it has to be precise. So, let us kind of imagine the worst scenario. The critical application and handle this. So, let us look at the timing diagram of this kind of 2 outputs.

(Refer Slide Time: 21:38)



So, this is a scenario, so we have a very simple 3 state and 2 states. So, we have the clock and we have a start signal which is going high like that you know, it goes high after the positive clock edge and for the 2 clock period say it is there. So, now you look at the Moore machine okay or Moore output. So, there you see there at the beginning start is low, so definitely you know that this is s0.

Here also you know that the clock edge comes is still low. So, this is s0 and the next is s0. Because there is no transition when the clock edge comes it is low, so it is s0. But here you see when this clock edge come the start is 1, so the next one is s1 and you know that s1 is next clock edge without any condition transit to s2. So, if you look at the Moore output the states are s0, then s0, s1 and s2 okay.

Now how does the SOC output looks like, we know that SOC is 1 when the state is a in s1 okay. so, the output will come like this you know you have SOC is decoded from the present state and it will be like that. It will not match the period. Because there is a output logic delay, because output is decoded from the present state. So, there is TCQ delay of the flip-flop + the t output logic that is it this one.

So, this is tcq maximum tcq+maximum output logic that is a worst case gap you can have and it appears like this okay. Now let us look at the mealy kind of output we know that there the same

thing the state is s0 the clock edge come it is still 0 it is s0. And it is start as gone 1 and you know the clock edge and it goes to s2, it there is no s1 in the mealy output. So, you remember that you know there it was start is high it comes to a s1 then s2.

But here start is high just goes to s2, but it generates in s0 an output okay SOC is generated as a function of start. So, that is what we are going go to see here in the mealy case, the first one is the s0, second one is s0 the third one is s2. But our output now you look output is in state s0 and the start is 1 so, basically it is a decode of that and so, you kind of and both together. This is 1 and this is 1, and since start is going high here it will come with the delay.

Because of the decoding delay so, you have something like that you know imagine little more delay. Because start is going so, I will not shown that but this is delayed a bit. So, this comes like that you know it is very nice so, if you look at the advantage yes one state less okay good and output appears kind of much before at least a clock period before the Moore output looks good it response faster okay.

Now the game comes the question comes what is wrong with this you know here probably nothing get go wrong. But here you see the output is a function of this state and the input. And this input mind you has got nothing to do with the clock and this input is not synchronise with the clock. Because this input is our state machine clock and this comes from a hose processor and that processor has another clock.

So, this may be generated in relation to that clock which could be much higher than this. So, it this pulse can appear can anywhere and the question is what happens if the start come late like that you know. I have shown with the fainted kind of mark so, if it is suppose the start is coming very close to this positive edge. Then what happens is that the SOC become kind of glitch you know we said that there is no great timing restriction on a SOC in the sense.

That can be a narrow pulse but if it is very narrow may not be detected or this could be thought of as some other output not a SOC. So, since the input if the case of if there is a case where the input is not synchronise and at the output, you will get a glitch which may not happen in the case

of Moore kind of output. So, it is a bad news okay it means that it is a bad news as well as the good news it depends on the way you look at it okay.

So, there is a famous story of a famous shoe company sending the sales man to a country where people you know long back people were not wearing the shoes or any footwear so, this marketing person was send to that country. So, naturally when round and your size were on the feet of the people. And he wired back know that time there is no internet and the mobile phone.

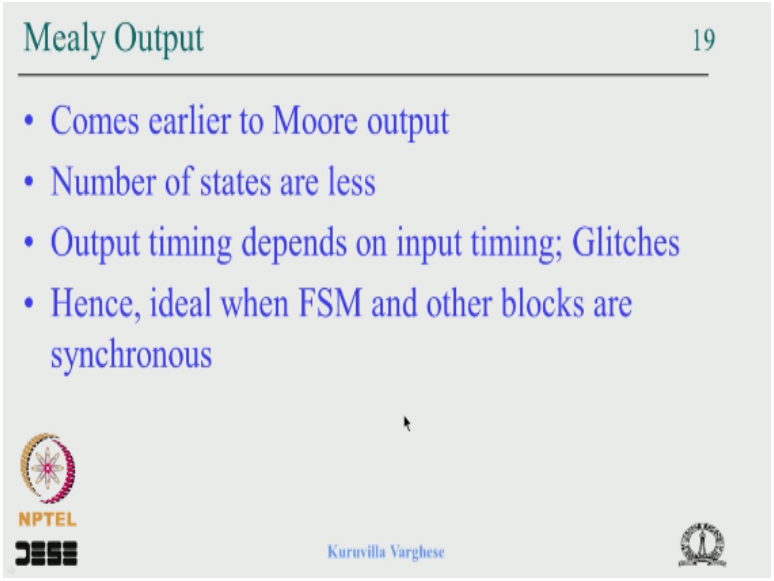
So, he wired back he send a telegram saying that I am coming back the people here does not wear any footwear okay. So, he went back after few kind of month another marketing bright young man came as a marketing person, you are sent to the same country okay. So, he went around looking at the feet of the people. He wired back you send two ship load of footwear because the people here do not wear footwear okay.

So, people not having the footwear could be bad news for somebody and good news for someone so, it is a way you look at it. So, the same situation here in our mealy Moore output the fact that the mealy output when in the presence of asynchronous input can produce glitch is a bad news. But it is a good news it means that when the input is synchronous, you can generate the mealy output.

That is what we should we learn positively from this suppose if this start is coming synchronise, synchronous would mean that in constant face relation to this clock. That means somehow this is generated you know indirectly from this clock so, that every time it comes with a specific delay with the respect to this clock edge. You know it need not be that it comes every time with the very near to the clock edge.

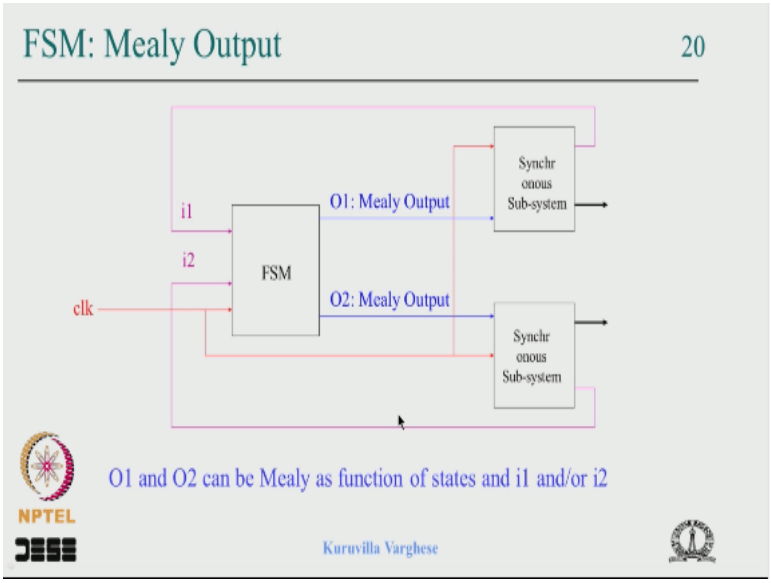
It can come every time with a fix face relation to the clock okay so, that is a synchronous so, when the input is synchronise there is no timing issue and the output can be the Mealy output.

(Refer Slide Time: 29:10)



So, that is the game that is the good kind of inputs.

(Refer Slide Time: 29:13)



So, I am putting that in the picture so, you have two synchronous sub system so, you can imagine this is the register which is kind of the control signal is given by the state machine maybe an enable signal or this is a counter which is kind of enable by the state machine. We have seen that or a load of the state machine counter goes from the state machine. And the state machine and all the synchronous sub system working on the same clock.

So, and you see that there is some output, we do not know what kind of output maybe if it is a counter, it is a timer, it is a decoded output which is going to the state machine input and so on let

it be anything, but it does not matter. So, this output is synchronous to this clock and that goes to the state machine. So, state machine in this case has 2 output i1, i2. It has 2 control output which is o1, o2.

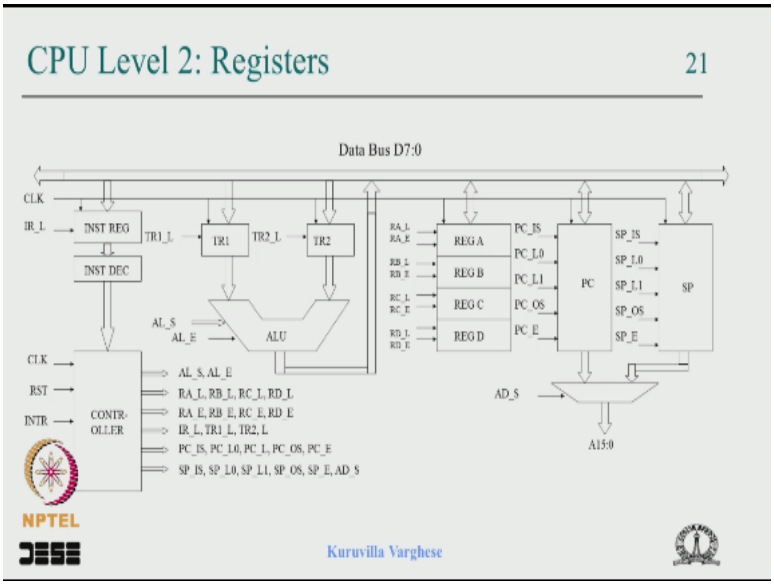
Now it is very clear that you know o1 can be generated as a function of i1 and or i2 okay it is a mealy output, o2 can be a mealy output as a function of i1 and or i2 okay. It need not to be that the o1 is generated only as a function of i1, o1 can be a function i2, i2 or i1 or you know there is yet another input which is synchronous to this clock it does not matter maybe it is coming from the outside world somebody has synchronise to this clock.

And that can be coming to the input of the state machine. Then this o1 can be generated as a function of i3 which is synchronous with the clock. So, that is a good news that whenever you see a kind of the synchronous inputs you can happily generate the mealy output as a function of that. If it make sense, it is not that you struggle hard if it is nice if it works out it is part of the spec then you can do that.

And that give you all advantage of you know the all these advantage. It comes the output comes early to Moore output number of states are less yeah glitch does not come since it is synchronous. So, that is all about the mealy output when we come to a case study I will show definitely 100% how the mealy output can make the life easy.

Only thing is that you should kind of from the day 1 when you practice you should try and think whether the particular output can e mealy output, that is my advice to you**.** So, that is about the mealy output let us move forward okay.

(Refer Slide Time: 32:01)



Now let us consider the little with this state machine generating the control signal. We have looked at it particularly we have seen an example in this case we have taken a register in our CPU example I hope you remember this was a the beginning of the course we have you know we were looking at the advance digital design and we have taken the CPU as an example.

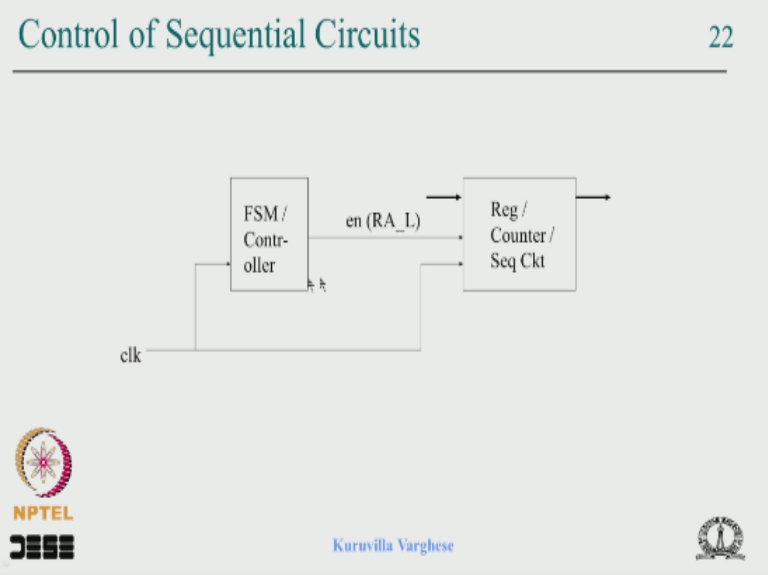
We are looked at the top down design, I have illustrated all the process, all the methodology by taking this example okay. So, there we had the CPU registers and it is an 8 bit register. And we said that input is connected to the register the q output is connected to the same data bus. Because there is only a single internal data bus and that was connected to the bus using a tri state gate. So, the game was that when some data from the input has to be latched to a.

The state machine will give a signal latch signal or an enable signal this case we call as RAL that comes here. When this is high and the clock comes the data was whatever was here gets latch okay, and we mention that it is not that this latch signal is going as a clock. Because it is possible that we want to kind of continuously load the load in some application okay not maybe the CPU register. In that case it is very convenient keep this latch signal or enable signal high for say 10 clock cycles.

So, continuously each clock the data gets latch maybe this is a FIFO or a counter which is getting incremented whatever but may not make sense for the CPU register okay. And we have seen we

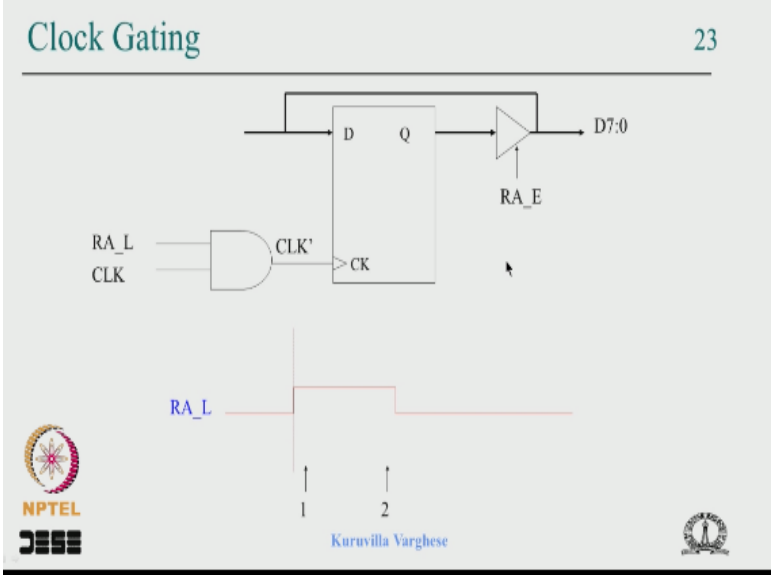
were looking at what kind of how the design should be to make that happen, that means when the latch signal is high and the clock comes the data gets latch.

(Refer Slide Time: 34:12)



So, every so, that is what is shown here. We have a state machine and some register or counter or some sequential kind of a element which is working with the same and the state machine is giving an enable signal here. So, that the input gets latch in this register okay.

(Refer Slide Time: 34:32)



So, and we have seen one possible kind of implementation very nave the very first thing which comes to mind. Because we say that when the latch is high and the clock edge comes the data should get latch okay so, this shows a 8 flip-flops input and output are combined through a tri

state get because there is a only single data bus and this is the enable signal and we have just combined okay. Now at the time of discussing this we did not look at the timing okay.

Because we probably we did not learn in a about the state machine now since we have learn we can look at the timing. Now assume that this is coming from a state machine so, the pulse is like that, but then we know that the relation of this pulse to the clock of the state machine. So I am putting the clock frequency the clock waveform so, we know that the state machine something happened to the state machine change as a state.

And the output is decoded so, there is the output kind of come delayed with the respect to the positive clock edge. So, there is a tcq the state flip-flop delay and TOL output logic delay and it comes here okay. Now you can already see that trouble that we are here so, you and this you see there is a overlap 11 here, and there is an overlap 11 here. Because of this AND gate delay it gets comes delay it, and so, you something like that.

You know get two positive clock edge and you see the trouble okay, we are expecting it be clock by 1 and it gets clock by 2 okay, clock twice and in the case of register you may ask what is wrong with it. Because say some data is here you get you clock it, it latch here you clock it again it may latch here okay. But look at the timing when this data may be coming from some other register in the ass usual in the data path and through some combination circuit.

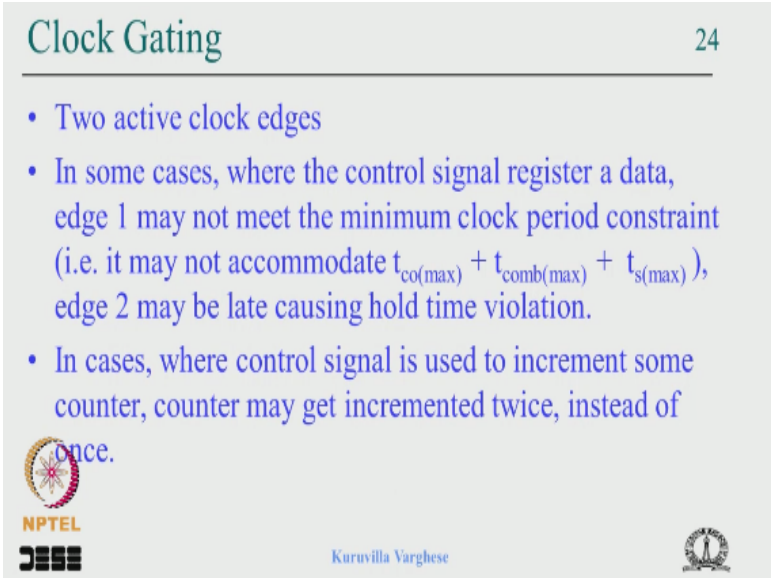
So, that is that means that that is also clock by the same clock so, data is kind of coming out of that register, and normally you know that we should be latching it t this edge. It is too early for like you know we have a clock edge here but the maybe that is too early, because we decide the clock period depending on the tcq tcom and tsetup. And we accommodate that in a clock period so, but here if you see the timing available from the clock edge to this clock is only little.

So, it is too early for registering maybe this is write like we have decided that the data appears here. But then you know that this the data gets latch here, but we do not know whether the data will remain there. Because that register is clock by this already the output is changing may be

this is coming in the hold time after the hold time into and so on. Then a wrong data can get latch.

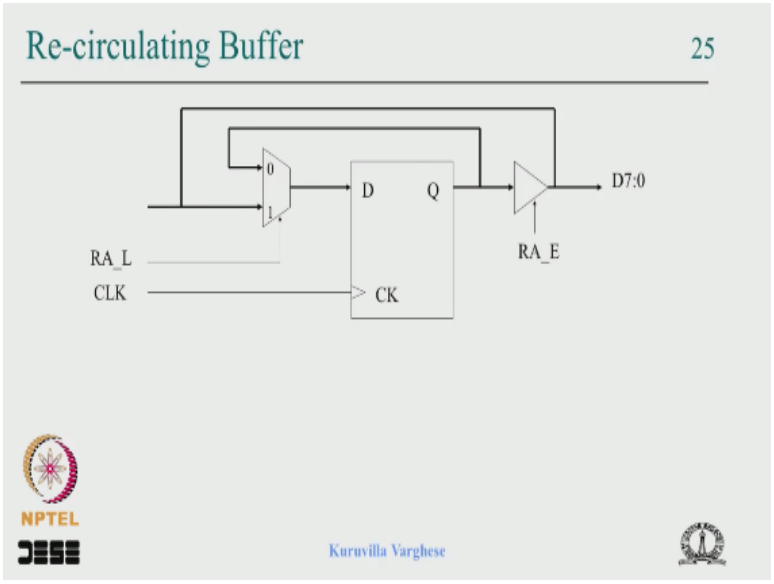
Suppose this is a counter which is getting incremented on this kind of latch signal. Then you know that in a edge comes it gets incremented probably again it gets incremented and so on. So, this is because of this glitch this scheme is not timing wise very kind of neat okay.

(Refer Slide Time: 38:15)



So that is why we have I have discussed these points.

(Refer Slide Time: 38:19)



That is why we have looked at this scheme okay, when we discuss the CPU I have shown this scheme and this does not have the timing disadvantage okay. So, here we are doing anything with the clock period clock path, we are combining the latch signal in the data path it has advantage in two ways. One is you look that when the latch is 1, the input is going there okay. So, input gets latch and you see that we know that because of the state machine decoding.

It comes delayed and it is made 1 and the input is coming from some register clock by the same clock so, it appears start appearing here. And it has all the time till the next clock period upon the cock, this clock edge the data gets latch here. So, it is very clean okay but only problem as you see is that the clock is coming every clock this is getting kind of clocked okay. So, maybe the data we enable sometime and the data goes here.

But all other time this same output is re-circulated back to an input and so on. So, assume that say in a case whether the latch signal comes only one in an hour. But unnecessarily this whole thing gets you know keep on switched so many times, suppose it is 1 gigahertz for 1 hour nothing happens. But the because of the switching lot of power is getting dissipated so, the re-circulating buffer is timing wise is very kind of nice very clean.

But the participation wise it is really hopeless, you know absolutely hopeless it has no relation to when the latch signal is high all the time it dissipates power it just a matter of frequency whatever is the frequency it gets switched. And the power is dissipated now so, it means that for the power as for as participation is concern the earlier scheme was nice. Because when the latch signal come you get 2 pulse s okay maybe clock twice.

But if there is a 1 hour delay nothing happens to the clock and there is no power dissipation. So, let us probably if you want low power dissipation so, when you design you can adopt this scheme and we have seen that we will I will show examples. But you are into the like low power domain where you are working with the handle devices, mobile devices then you need to dissipate less power anything better you operated or even now it is wise.

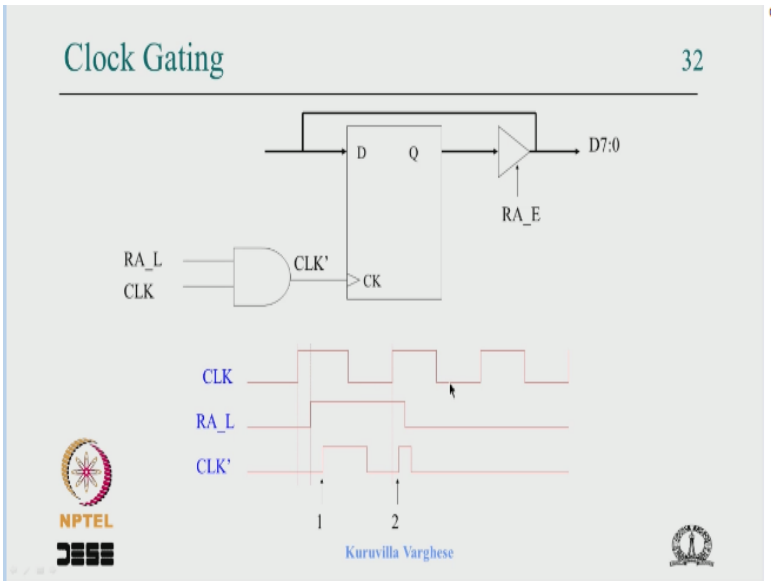
You know there are even in a desktop PC there are millions and millions of PCs all around world even slightest you know power saving will kind of a help to reduce the overall power consumption of the world. We should always look to reduce the power dissipation so, here see the game that trouble comes because like we need actually the hour kind of clocking should happen at this edge. Because when a latch signal comes here.

The right time to clock is this second edge because the data has changed at this clock edge and it has to propagate and come here. So, the right thing do is that we would like a pulse like this okay definitely but the problem is that because it is coming you know half into this clock period it is problematic suppose this kind of this latch signal instead of coming with respect to the positive edge assume that it is coming with respect to the negative edge.

So, this pulse will come you know start like this and stop like this and if you and it then you get one pulse very cleanly here okay. So, that is the trick okay so, if we can resynchronise this latch signal with the negative edge okay. Then it comes like this and you can with the resynchronise latch signal the clock is ended with the resynchronise latch signal. Then you will get a pulse correctly like this and you can use it to clock this particular thing.

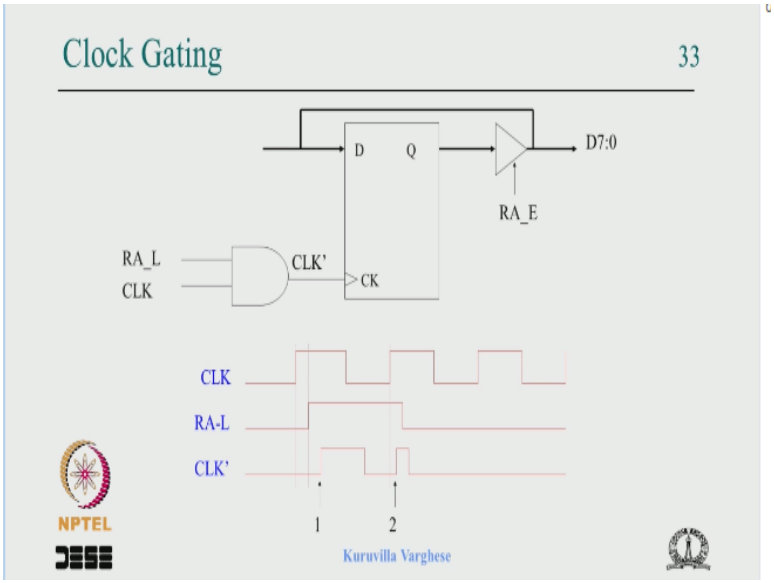
So, I have explained what is the idea behind so, that is what I am going to show now. So, maybe I will show that and come back okay.

(Refer Slide Time: 43:26)



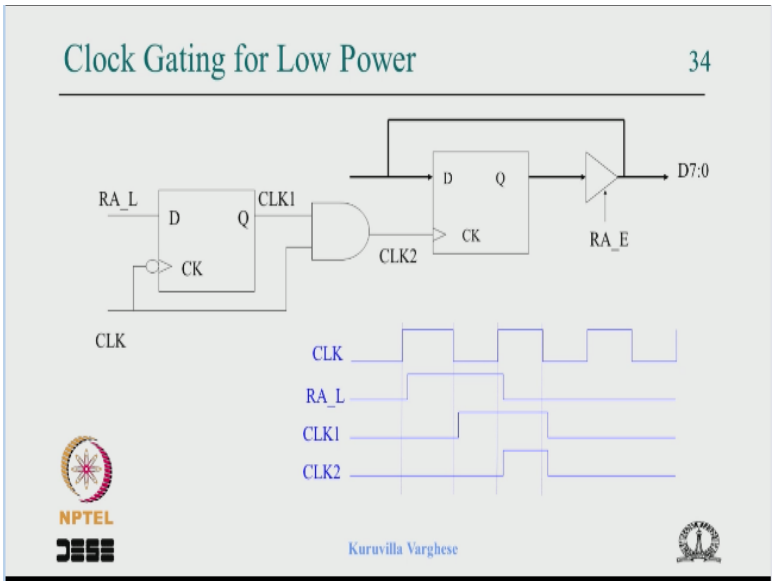
So, this is what is if we were discussing, so we will what we will do is that we will re synchronise this with the negative edge it comes here then and it okay.

(Refer Slide Time: 43:45)



So, let us look at that you know, so that is what is shown here.

(Refer Slide Time: 43:48)



So, you have the latch signal which is coming from the state machine. Now we put a flip-flop and there is a clock which is coming and it is a negative edge to gets flip-flop okay, you see that there is a bubble and the, so that means this latch signal is re synchronise to the negative edge. So, that is we have clock 1 and that is handed with the original clock and which generates a clock to which is used for clocking okay.

So, let us put the waveform, so you have a clock and normally the latch signal comes like that and then we are looking at the clock 1 here. Because it is going to a negative edge to get flip-flop. So, it is see when the negative edge comes this is 1. So, the clock 1 output is 1 and when the negative edge comes this is 0, so it becomes 0. So, you get a pulse with the delay with respect to the negative clock edge.

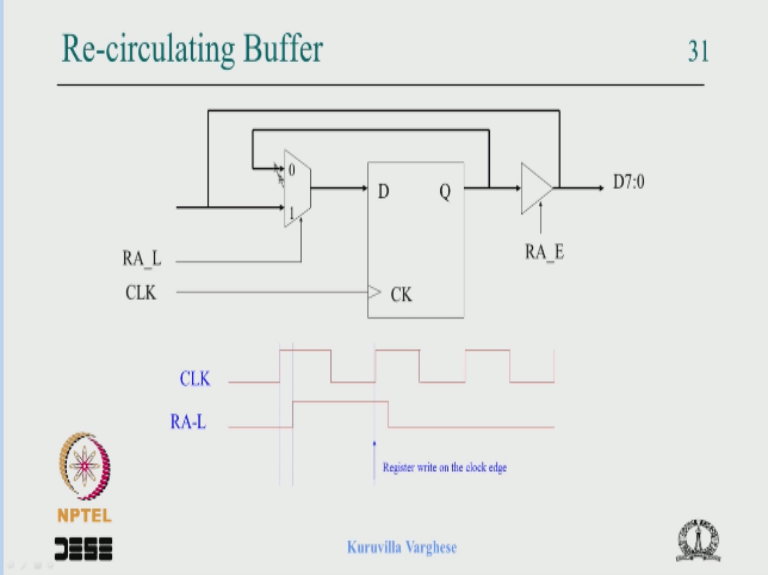
And you and now this clock 1 with the clock. So, this clock 1 and clock is handed and you get a clock 2 with a delay, little delay. And you know now you are correctly you are something is happening before at this edge. And we are supposed this edge and that is kind of latch with this particular clock edge and everything works smoothly and there is only one pulse .This is clock and it does not continuously dissipate power.

So, when you need to kind of you need a low power the clock gating, then 1 should adopt this scheme. And it is very standard scheme know earlier we had this scheme and this is converted to

this. So, what many a time tools do is that when they see such a clock gating. They can easily convert automatically to this by inserting a kind of negative edge to get flip-flops.

So, many a times tools try to automate this kind of thing looking at this scenario it converts into this.

(Refer Slide Time: 46:08)



Even it is possible that you implement with a re-circulating buffer and the tools detect that it is a clock gate kind of re-circulating buffer. And it can do the proper clock gating by you know taking this re-synchronise with the you know it can just rephrase that with is, because very regular and that is kind of very close to the it has to be very close to the real flip-flop okay. otherwise there will be timing issue.

Because you are playing introducing this q in the path of the clock. So, this is many a times done automatically by the tool you give either this or this. Then that can be kind of detected as a case for proper clock gating and that can be introduce and the FPGAs can be built with kind of re-synchronising flip-flop near to a register and so on. It depends on the FPGA. So, when we will look at the FPGA architecture.

Then we will see is there a scope is there a flip-flop nearby another flip-flop. So, that this kind of game can be played, but mind you there is 1 thing I should be mentioning here. So, it looks that

sometime you know it appears that it is enough to avoid this glitch problem if you kind of push this gate signal you know the enable signal here you know by introducing certain delay you know.

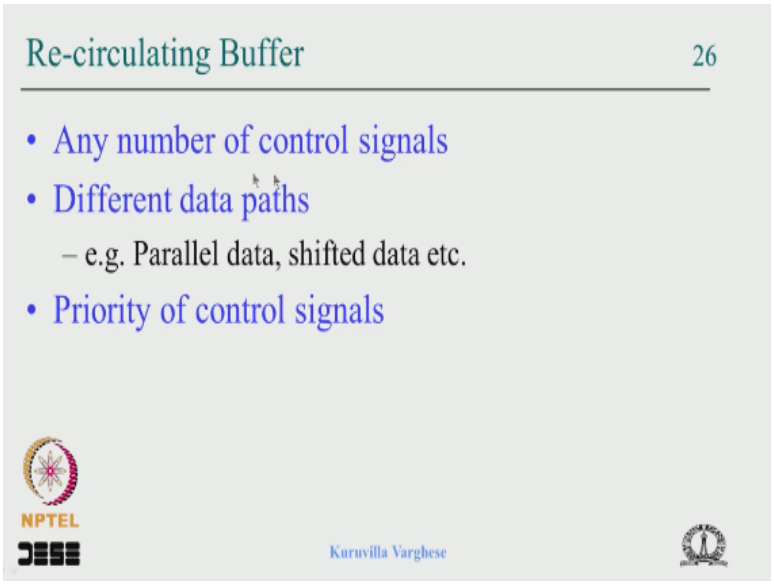
You introduce suppose you can introduce 7 nanosecond and it appears here. So, you might think that you know you can add a kind of two inverters to get the delay but the you know that this delay depends on the clock period. And if you add arbitrary delays and change the clock frequency does not work, and you know that the delays are function of the temperature and the supply voltage and so on.

So, if you kind of in the slowest path, you introduce the delay and when it becomes first, this may not work and so on. So, that is why resynchronise with the negative clock edge that you should keep in mind so, you should not try to kind of delay this. And kind of you know gate it you know that does not work, because of the reason mention and now this is what we have seen.

So, this at least we said that this also problem of kind of the participation. But many a times we use the same technique to achieve not only one in this case it is just a latch signal. But there could be any number of kind of control signal in the data path that we have seen in the case of a counter it not at just one control signal. This is just like an enable signal for a register, but assume this is a counter this could be an enable signal from the state machine.

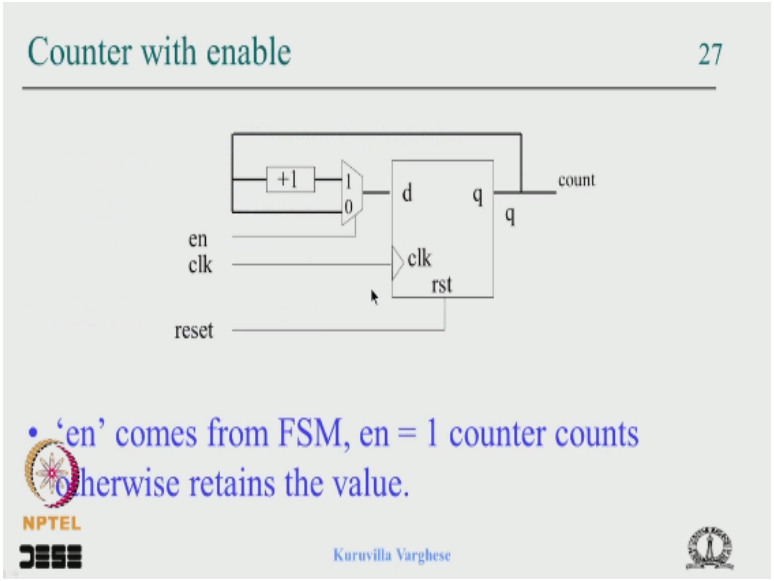
There could be a load signal from the state machine there could be an up down signal from the state machine and so on. So, this scheme can be kind of extended we have already seen it. But I just maybe it is a write point to stress it again.

(Refer Slide Time: 49:32)



So, you can have any number of control signal you can have a any number of control signal, you can have different data paths like parallel data, shifted data. And when you have multiple control signal, you need to have some priority like if both come together which takes precedence okay. That should be kind of decided and that is very natural we have see that control signal which comes close to the d will have the precedence.

(Refer Slide Time: 50:02)



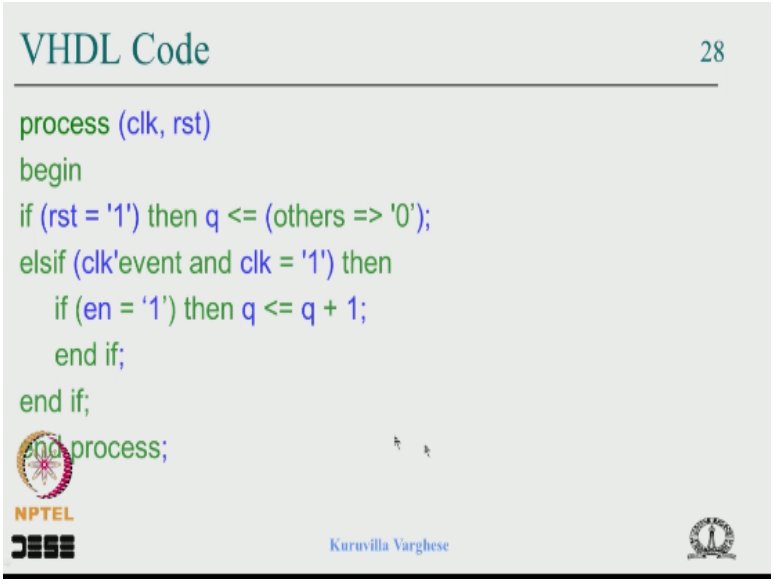
So, let us look at this you know let us look at this example we have already seen it. So, this is a counter with an enable okay very useful kind of structure so, the clock suppose this is a say 4 bit counter, then you have 4 flip-flops. The cock is common, reset is common and suppose this state

machine has an enable it is at the reset at the beginning. So whenever it is enable it is incremented that is the basic idea. So some event happens the state machine enables it.

And it gets incremented you know that is greened game. So, how it is implemented you have a 2 to 1 mux the enable is this select line of the mugs when the enable is 1, q is nothing but q+1 and incremented is in the path. And otherwise the q is re-circulated so, definitely there is participation but the timing wise it is very neat. And we know how to write a VHDL code for it so, we write a process. Because we know that is a single process will work for registers proceeded by some combination circuit and this is what is the combinational circuit.

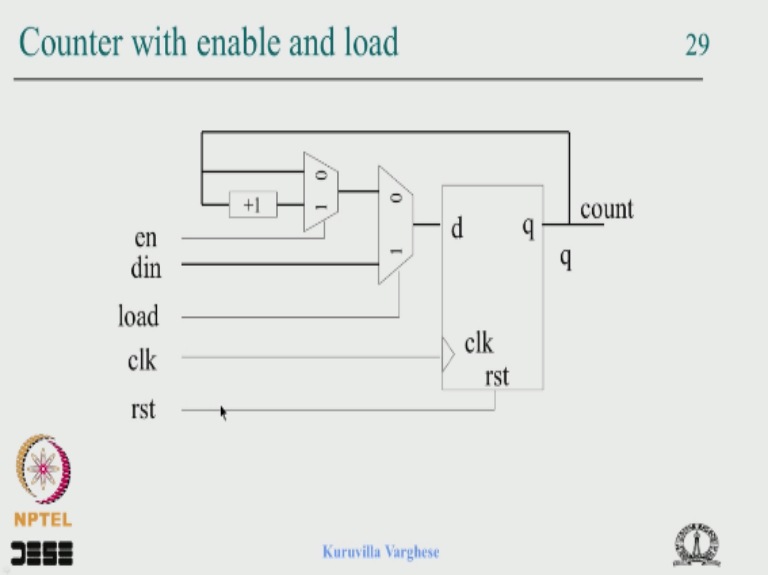
So, we say we write a process with clock and reset in the sensitivity less, and we say begin if reset is 1 because it has the priority asynchronous if reset is 1. Then q gets others 0, else clock event clock is equal to 1, and this is synchronous we say if enable is 1, q gets q+1 end if that means else you re-circulated. And end the end if of the first if an end process

(Refer Slide Time: 51:43)



So, that is what shown here process clock reset begin, if reset is 1 then q is others 0 else if clock event clock is equal to 1, and under that because there is a control signal. If enable is 1, hen q gets q+1 end if okay, that means re-circulate if not q is q that is the meaning of it. So, that is a 2 to 1 muxes which is synchronous which is coming to the input of the input d of the registers.

(Refer Slide Time: 52:12)

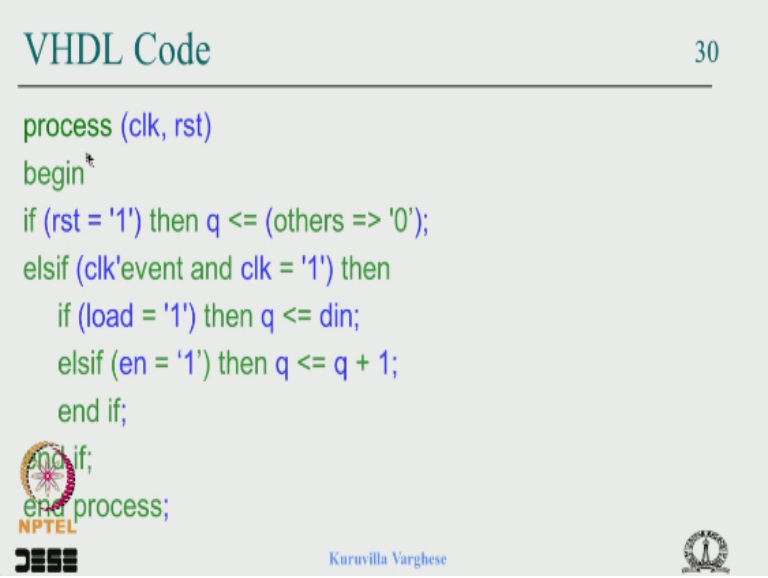


So, let us take another example where there are not 1 control signal 2 control signal. We have already seen it but just reinforcing the concept so, here you have a load signal, a 2-1 mux and when the load is 1. The input which is 4 bit gets latch here upon the clock, and if the load is 0. Then if enable is 1 okay so, it has 2 control load signal and enable control. Enable is 1 the count gets incremented otherwise count gets re-circulated and naturally.

We know that this as the priority because this comes first so, if respective of the enable. If load is 1 the input gets loaded when the load is 0 then depending on the enable either it is incremented or the output is held okay. And we know once again this register with this combination circuit can be coded as single flip-flop, this is synchronous. So, this comes it you know within the clock event clock is equal to 1.

So, same thing process reset and clock begin, if reset is 1 q gets q, you know q is others 0, else if clock event clock is equal to 1.And the highest priority load is 1 q gets din, else if enable is 1 q gets q+1 end if okay end process you know .

(Refer Slide Time: 53:37)



So, that is the process clock reset begin if reset is 1, then q gets others 0, else if clock event clock is equal to 1. If load is 1 then q gets din else if enable is 1then q gets q+1 end if that means else the q gets q, this is the end if for this. This is the end if for the main thing and the end process okay. So, that is what we have I think we have come to the last part of the lecture before taking up a new portion.

I would briefly tell what we have done we have looked at the mealy and Moore output and we have seen that Moore mealy as an advantage number of states are less. The outputs comes early but does the problem of glitch so, it works neatly with the synchronous kind of input you can straight away go for mealy output if it permits and we seen the control signal and in the specially in the case of register a clock gating has time issue.

Because there are tqo pulses and the solution is that to go for re-circulating buffer it does not have any problem with regard to the timing. But all the time it is dissipating power and this can be extended to the multiple control signal we have seen a case of a counter with an enable. And where it works kind of without any timing issue and we have seen the VHDL code and when there could be a also seen a counter with load.

And enable with which load as priority and once again it can be coded in a single process and we have seen the example we have seen it earlier. And we have also seen a kind of low power

solution of clock gating to avoid this kind of glitches. We resynchronise this latch signal with this negative edge and a adding a constant delay want work. Because the clock period can change and the delay of the combination circuit itself can change.

So, we resynchronise with the in the negative clock edge and the resynchronise version of the control signal is gated with the clock. And that works is clock so, this works very neatly there is no issue with the timing only thing is that it uses an additional flip-flop for a set of register not that it is there could be 16 kind of flip-flops. And one additional flip-flop is required depending on of per control signal.

We can say we require an additional flip-flop for each control signal to resynchronise that is the basic idea of clock gating for low power. So, today what we have seen is that we have completed the mealy and Moore output and we have seen the advantage where the mealy can be safely use and we have looked at the clock gating solution re-circulating buffer, what is the issue timing issue the clock gating and how it is avoided in re-circulating buffer.

But the issue is that the power dissipation and we have seen a clock gating scheme for the low power and we have seen the extension of kind of re-circulating buffer for multiple control signal whether there is priority how to design it, how to code it and all that. So, in next lecture we will try to take some more issues timing issues and other issues of the state machine.

Then I am hoping either take the VHDL coding of the state machine or we probably start looking at the PLDs. So, that is the plan so, today I am winding up I advise you to go back because we have covered quite a lot go back and review learn well, I wish you all the best and thank you.