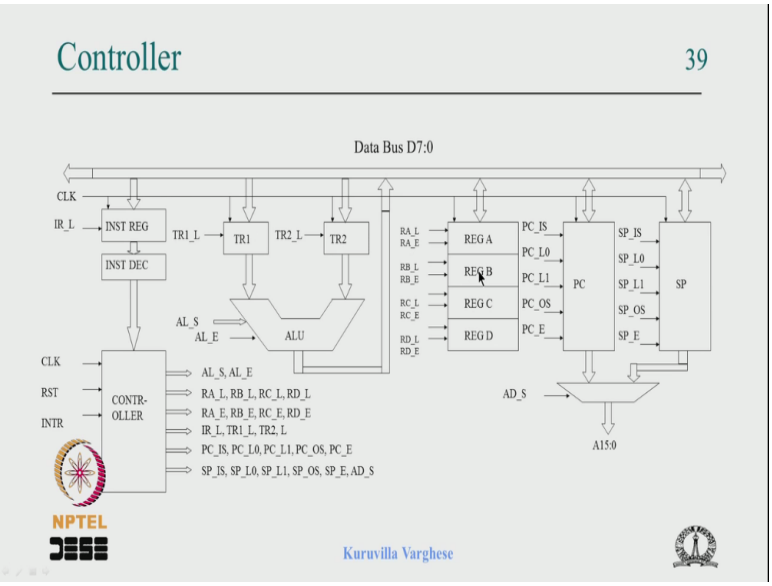
Digital Systems Design with PLDs and FPGAs
Kuruvilla Varghese
Department of Electronic Systems Engineering
Indian Institute of Science – Bangalore

Lecture-07
Control algorithm and State diagram

Welcome to this lecture of the course digital system design with PLDs and FPGAs and the last lecture we have seen the behaviour of the controller and the structure of the controller how to design the controller and so on. So before going further about the some timing analysis and design let us run through those slides very quickly.

(Refer Slide Time: 00:56)

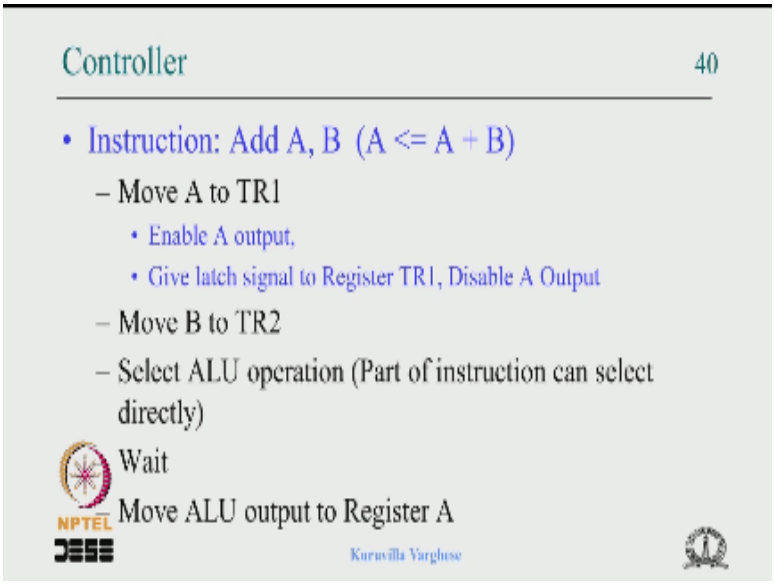


We have seen in the slide an example of the CPU and as I said the controller job is to give control signal to this data path and we have taken an example of instruction adding in A and B and putting the result back to A, the execution part of it, but that is enough to kind of enough for analysis. So essentially we moving A to this register, B to this register and selecting the operation and moving there early output back to the register A.

Now that each moment involves, enabling giving a latch signal and up on the clock it moves and disabling. So this is achieved by giving a pulse same pulse to enable a latch with respect to clock in or you have a certain timing with respect to clock and then when the clock comes it is transferred then it is disabled. Same thing to be B enable latch and the clock comes it is transferred and disable.

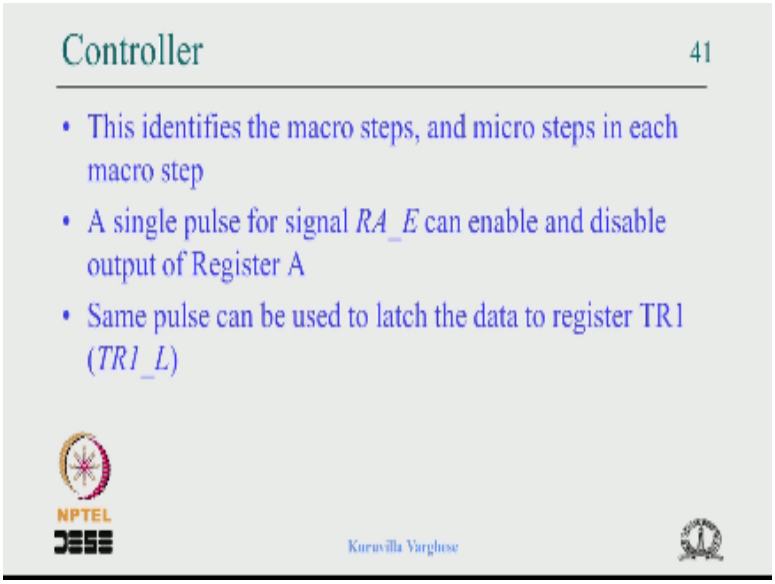
At the same time give the addition code here, as I said there are various units to multiplex output comes and Tri-state gates are there, the output of multiplexer, the enable signal of that is this one. So you give this enable signal, A latch and the clock comes this transferred and it is disable.

(Refer Slide Time: 02:26)



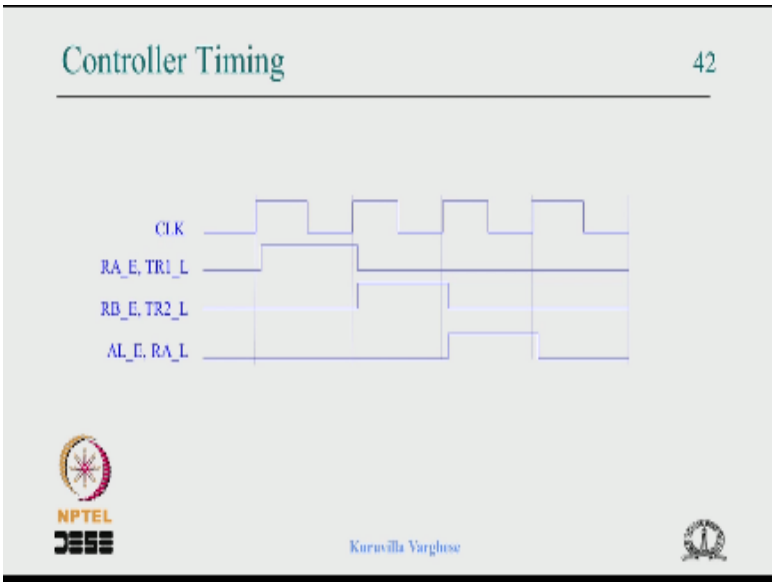
So that is what is there move A to TR1 which has a macro stuff which is enabling a giving large signal and the clock comes to transfer and disable A, everything is done moving be Tr2 selecting any operation and then moving A back to the register A.

(Refer Slide Time: 02:39)



So basically I said that the possible timing waveforms are from here, enable and latch, corresponding latch get the same signal, A enable this one latch B enable to latch, ALU enable and register a latch and as I said this waveform itself is a pulse.

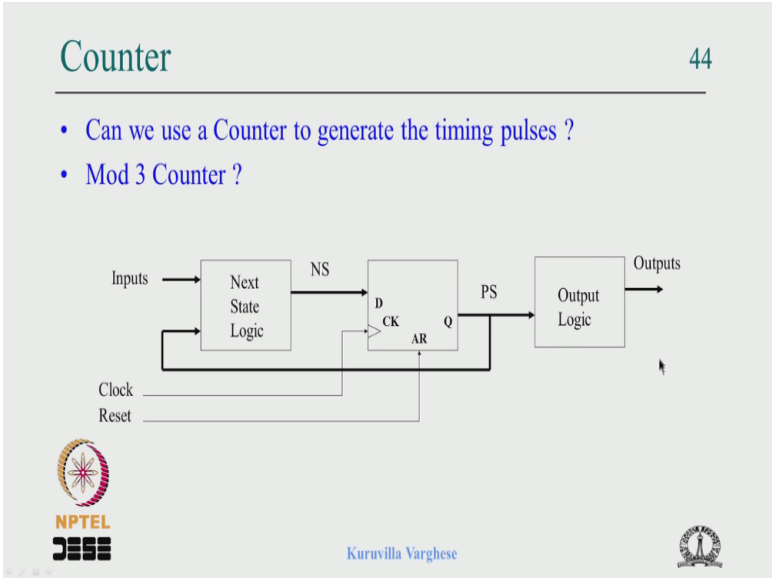
(Refer Slide Time: 03:06)



So the first duration of the clock, second duration of the clock and third duration of the clock, it is little bit shifted as if it is a kind of cooking up the waveform, so that the clock edge comes when the latch is high and I said this looks like little bit manipulation at this stage, but we will see how in practical circuits it comes very naturally and it works effectively. So the question is that how can we generate these pulses, what kind of circuit can be used.

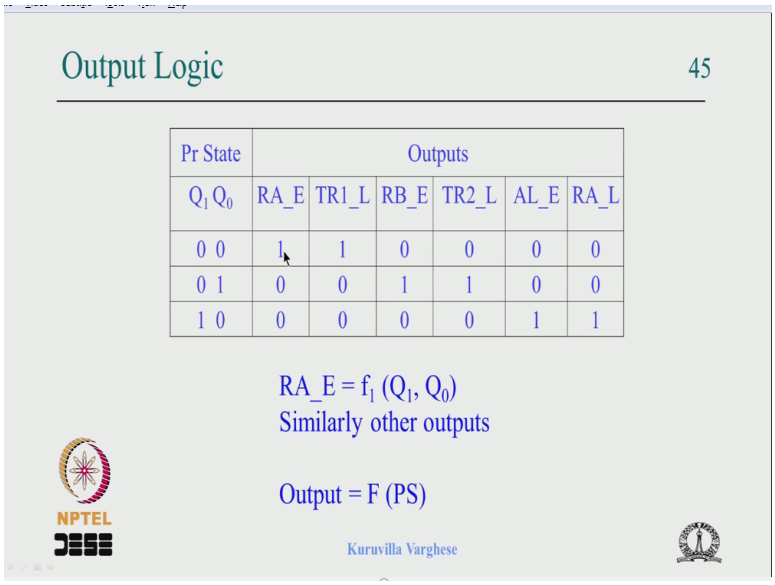
Definitely this cannot be from combinational circuit because some input like add A, B code gives to combinational circuit cannot produce a sequence, it has to be a sequential circuits and the question was that whether we can use a counter to generate this. So assume a counter counts every clock period like 0, 1, 2 then it is enough at the 0 count a pulse is generated for these two signal separately.

(Refer Slide Time: 04:23)



At the count one this is generated at the count 2, this is generated, at least for this part of the game Mod 3 counting is enough so that is what we have shown you have a mod 3 counter with count 0, 1, 2 you put a logic here to look at the present state of the count and decode like when it is 0, 0, 2 signals review on when it is 0, 1 other two signals will be 1 and 4.

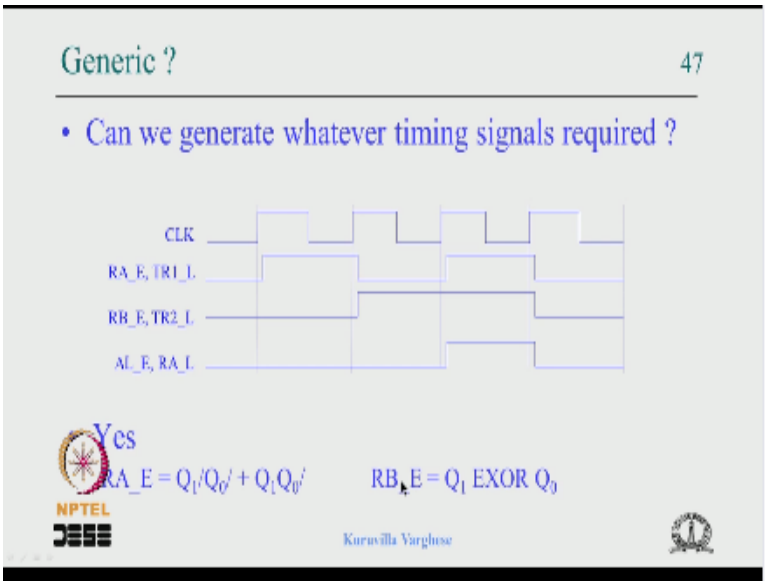
(Refer Slide Time: 04:48)



So we write the truth table of this. Then you like the input is the present state Q1, Q0, outputs are these, these setup signal definitely this has to be repeated ok, for convenience I am showing it together but assume there is a separate column for A enable and TR1L, it is not tied together but this happens only for this instruction, maybe some other instruction it not be together but just for the convenience have shown like that.

And so here this particular signal is a function of the present state. So we say output is a function of the present state. So we decode we put an output logic and then decode output of a function of the present state that is how we generate this pulses.

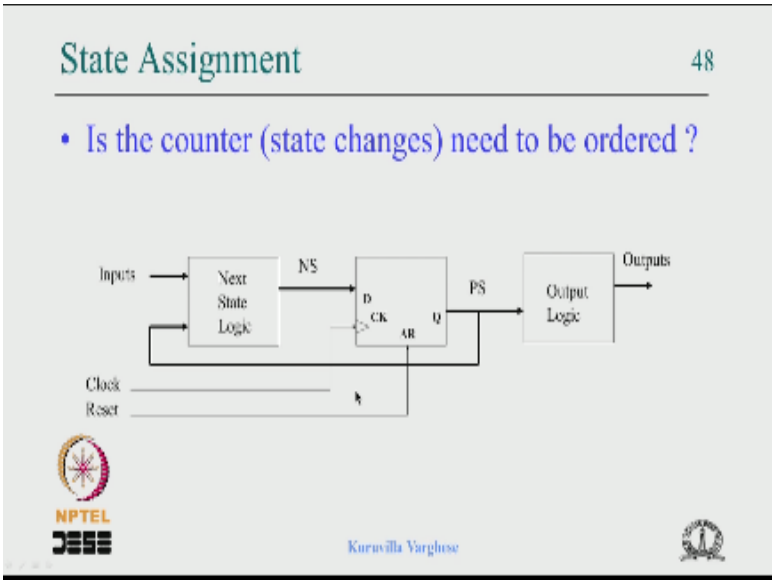
(Refer Slide Time: 05:42)



Then the next question was that is it generic that you know you have some crazy pulse requirement for the controller to work, then this structure can I produce, have shown some example, not maybe a realistic you know instructions waveform, but let us assume put some random thing and try whether it can work. So here you need a pulse in the first clock period and the third one, the second one you need second and third.

And in the third set you need in the third clock period. So as I said it is easy essay the first signal is nothing but decode of 0, 0 or decode of kind of 1, 0 and the second one is a decode of 0,1 and 1, 0 which when minimised you get you get Q1 EXOR Q0. So really does not matter that you just whatever you know the output logic you know is changed for various decoding to produce direction and output that is all what is required and this structure.

(Refer Slide Time: 06:54)



So is quite generic and it can be used for any kind of controller that means you design a controller for multiply or divide or some other any application. Then the same structure is used only thing is that depending on the need of these various transition, various inputs and outputs, the next state logic changer for the various transition and output logic changes for the various output in each step.

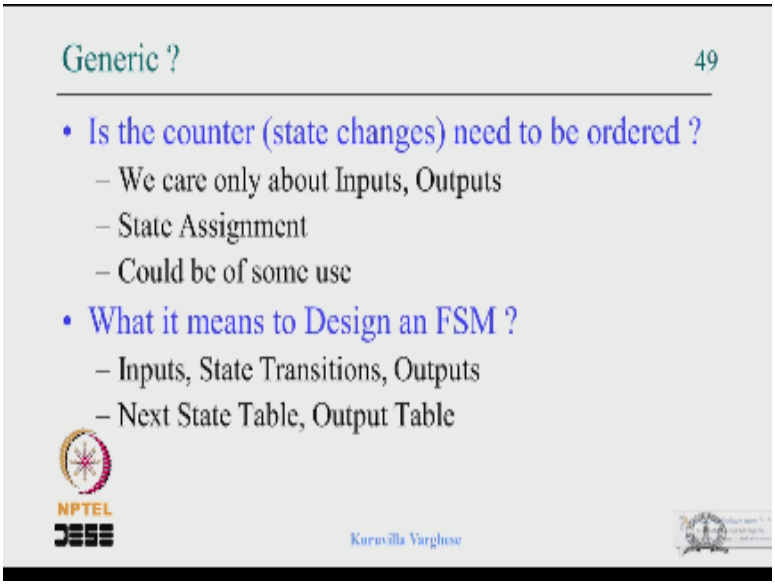
You know because that we have a sequence or stage this controller is going through and we produce that each state various output as and when required and the sequence itself is in terms of the clock period. So that is very important to understand everything is in terms of the clock period, the granularity of the controller is in terms of clock period. We like we can find the least of the smallest app we can do it is an clock period that should be understood.

And another question is that whether we do, do we do this machine need to count 0, 1, 2 we do not care, you know it can you count 2, 1, 0 only that the logic is change, so normally when we design a state machine we are worried about the inputs, the transitions and output and we are not very much worried about how this account counts you know and itself this is not a very in a complex case it is not a very sequential kind of state maybe depending on some input it branches to various state.

And their it for the branches to some state come back to locally come back to the original state and so on depends on applications. So the environment state transition can be very complex and so we do not care about the stare or nor we can assign some meaningful like we cannot start saying that 0, 1, 2 ,3 makes no sense. So that problem in assignment and the

controller is also called as I said finite state machine. Because it transit through the number of finite state for the whole operation. So it is called finite state machine or FSM ok. So that need to be you no understood.

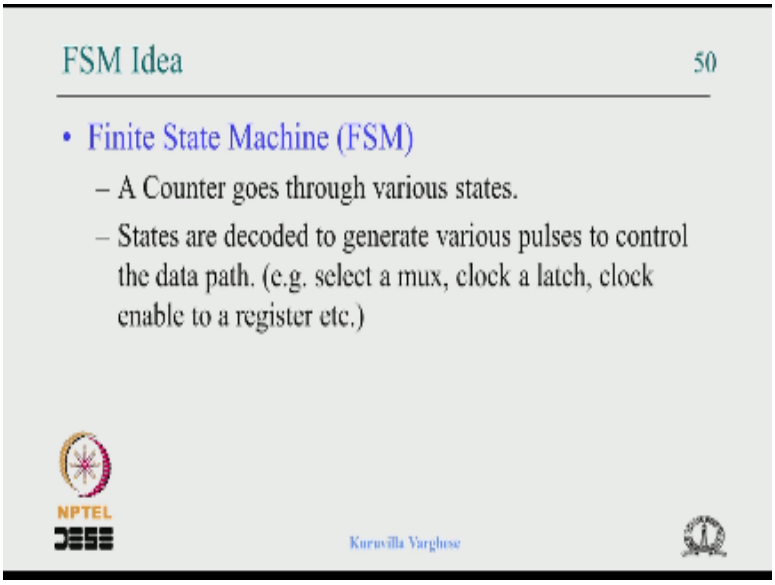
(Refer Slide Time: 09:31)



So and ultimately so we said that it is the counter and need not be ordered the state. So we have a step call assignment, so when to when you normally work we work on symbolic state at the end we assign numerical state and this can be manually done by the designer of tools can do maybe you can do state assignment in such a way that maybe this one of these logic is minimise, sometime it is used to solve some timing issues. All that we will see later as we progress and essentially when you are designing FSM we have some specification, some timing waveform.

And we translate that into basically some input, state transitions that means sequence of state and in each state what are the outputs, that is what we are going to the design and from there we can design the next state table and output table. So we start with the specification in terms of description and waveforms, then we work out these that waveform into this in input state transition and output.

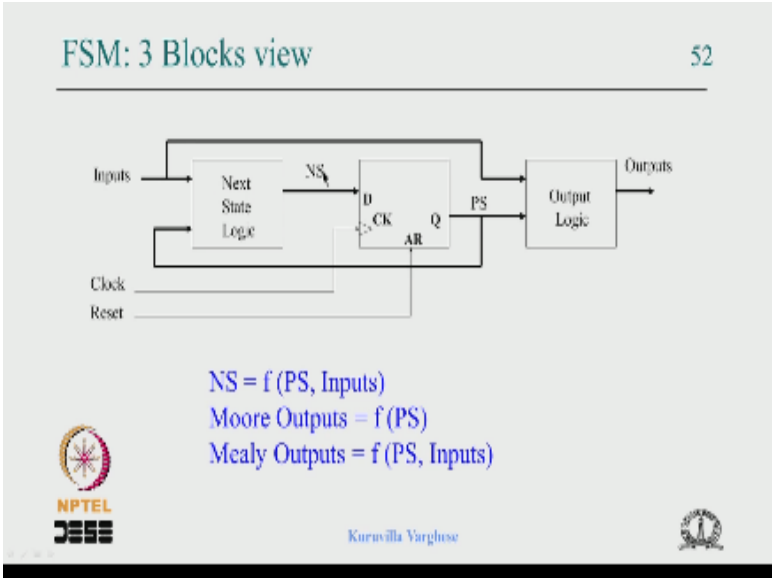
(Refer Slide Time: 11:00)



And ultimately we work out the next state table and output table, then the structure is known and we can design the controller. So basically that is what is the finite state machine or a controller there is a counterpart with me not sequential count, it goes through various states, various steps, various sequences and at each sequencing each step some output is you know decoded which has you know which does lot of function like enabling register and enabling counter .

Selecting some path to a Mux and all kinds of things. So that is the function of the controller as at the beginning of the lecture I said do not confuse it with the way you write a VHDL or Verilog code and say the financial aid machine do a comparison nor found or make any anything, it just make some decisions as to go to some particular state produce output to control the data path okay. Does not do absolutely no data of computation at all ok, everything is done by the data path combination circuits that should be understood well.

(Refer Slide Time: 12:14)



So we also talked about the Moore and Mealy machine and many a times like there is a possibility like we have seen the output is generated as a function of the present state, but in principle we can generate output as a function of the present state and input it is convenient it has some advantages in some specific cases it cannot be done just outright in all cases.

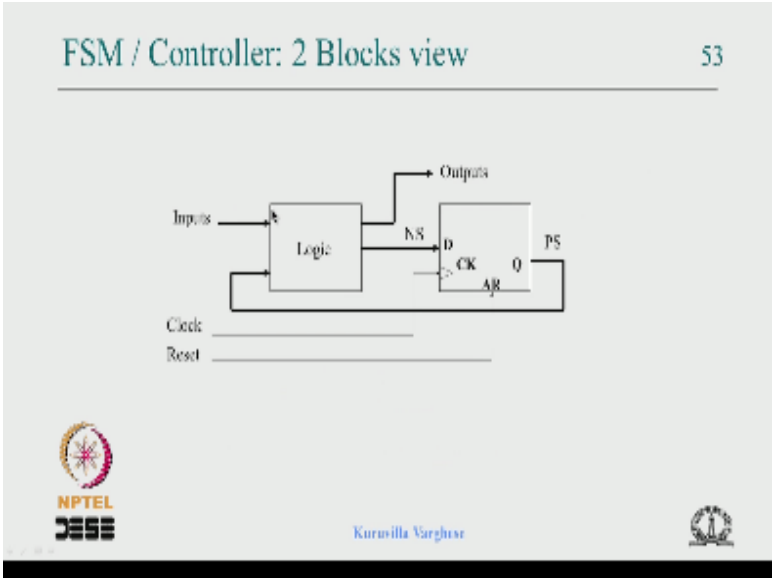
But in some cases which we will see where it can be generated, the output can be generated as a function of state many a times, textbook treat as 2 machine, one is Mealy machine and Moore machine. But as I said it is very like these are kind of text book pages with one input 1 output, so is very easy to categorise like that but in real life there a lot of input output.

So we talked about Mealy output when that particular output is the function of the present side and input and Moore output when the output is a function of the present state and input and we need 2 kind of in generalize that process of thinking in terms of the Moore and Mealy because if you get used to the Moore kind of output like working that in your mind then sometime in a miss.

And because the Mealy output could be advantages, so you should check the Mealy output can be designed. So let us come back to the slide. So this is the 3 blocks view as I said you have a state variable flip flop and next state logic is decoding, the next date from the present state and input and output logic decoding the present state and input. You should keep this in view be used for analysis maybe used for coding maybe use it for solving the issues to think.

And 3 block view is a good kind of abstraction and maybe as I said if you look at these two blocks both get the same kind of input, present state input and we could combine this as a logic.

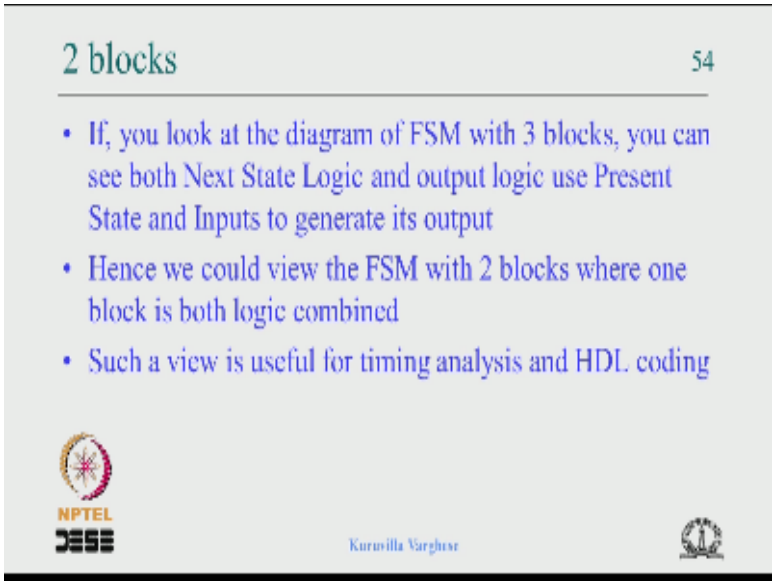
(Refer Slide Time: 14:255)



So that is shown here you have a logic which comprises of the next logic and output logic. So this logic in board design boat various inputs and the present state and there is a part which decode the next state, does a part which produces various output and this is also useful for analysis and the coding analog. So both should be kept in view the text book shows either some textbook show the early views, some shows this view does not matter.

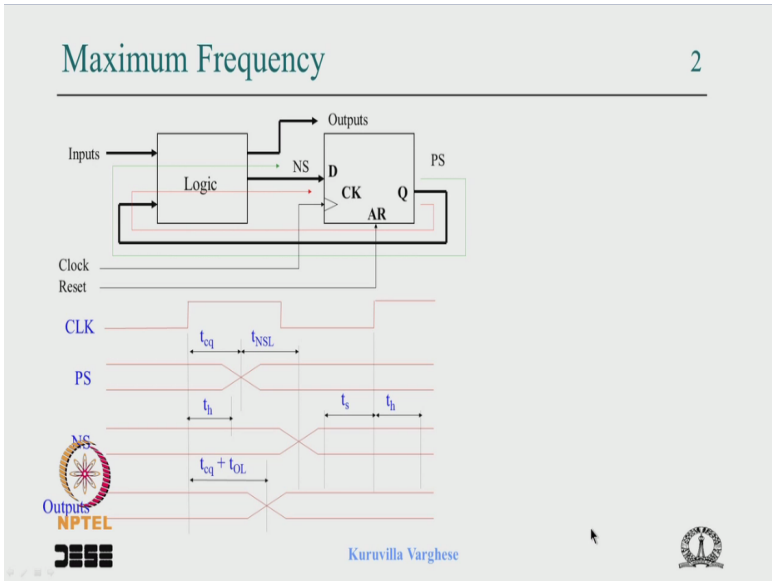
The understanding is important weather but this kind of pictures keeping that in mind will help to sort out the issues analyse and so on.

(Refer Slide Time: 15:11)



So that is what is you know picture here.

(Refer Slide Time: 15:14)



So let us come todays kind of part, so let us analyse the timing of the state machine and if you take this structure to block you is kind of easy to kind of analyse. So sure here because it is identical to the counter only one part is an output is an extra thing. So when we work out the maximum clock period there are two sets of path have to analyse, one path is same as encounter.

One clock comes the Q changes and kind of propagate through the next state logic and the next said come and get ready before the set up time okay. So that is what is shown here you have when the clock comes this present state changes after a TCQ time because from D to Q

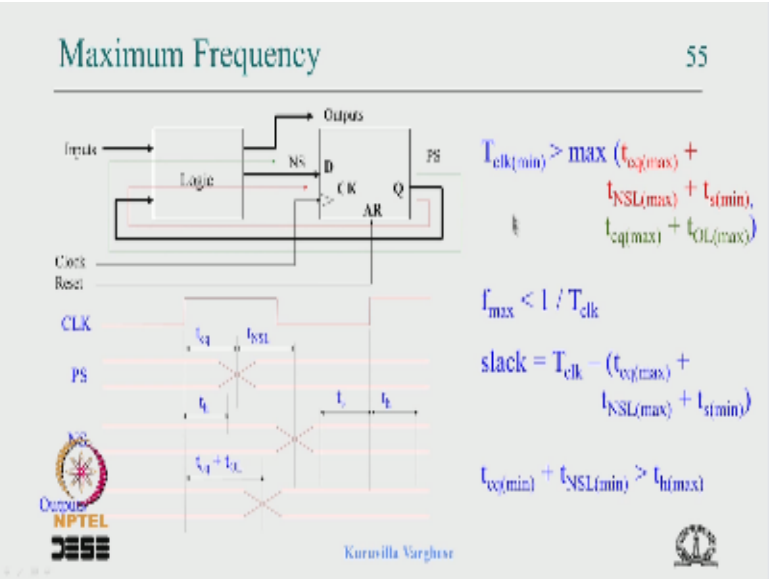
it take so much time to change. Then it propagate through the next state logic. So at this point the data comes here ok.

But we know that before the next clock edge comes the proper operation of the flip flop, this has to be set up before the next state, the data has to be there sometime before call setup time. So the cock period from this edge should accommodate the TCQ+TNSL is a timing the propagation delay for the next state logic and the setup time. So the clock period should be greater than the sum of the TCQ, tNSL and TS.

And of course be given more than it is not that even if you take the maximum it is better to give a margin and we many times does not worry, but there are so many changes the temperature change and the power supply changes and all that, because of the power supply voltage changes the timing can change it is always better to give a little slack, little margin. So that it does not violate the clock period requirements.

But here one other parties there like you at the same time in the clock comes use after TCQ the present state changes but for output it take some output logic delay and output changes here ok and it looks at least when you add some here we are adding up three times but here we are adding two times, but we need not like it does not matter may be the output logic delay can be greater than the next state logic delay plus the set up time okay.

(Refer Slide Time: 18:10)



So we have to consider the maximum delay of both the clocks like that is what I have written here, Tclock min is greater than max of two paths, the red path which is the red line here, this

path register to register or register to output okay. Suppose assume that you add up here, you get say 3 nanosecond and you add up this then you get 4 nanosecond and if you choose a 3 nanosecond for the clock period then the state changes happen properly but the output does not change, before the output setting the state is changing, output never get to change properly that there could be glitches and things like that.

So you have to consider this ok. Now once again Fmax is the less than 1 by Tclock and the slack is like you subtract the maximum path I have shown only this from the clock period but the maximum path from the clock period you get the slack. But one you should understand when you put a picture like this we assume the output is coming here ok. I like in this analysis means you do not know where this output is going.

So we have take treated this as the end point when you put such an expression, but in real life is a controller this goes to somewhere maybe this goes to a another registers and you know the data input or it goes to some combination circuit like a select line of a marks and that marks output goes to register. So in such cases for the completion of analysis you have to consider completely from this register through the output logic through the whatever is the components circuit following up to the next input of the register okay.

The complete then only will get the picture but when I saw the picture like that I have no you know affair knowledge of that that is why I am writing like this, but is there is a path terminating on a register, you have to consider the whole path, ok and in a sequential circuit definitely will terminate in some kind of register other than you know very simple kind of indication or something like that like a some lighting up LED or something like that.

Then maybe this analysis helpful, but that we should remember that go somewhere and when you analyse you have to analyse from this point all the way up to the to the end point that you should keep in mind and ultimately you have if you find you have together if you find that the clock period cannot talk right that is why I have to increase clock period or the frequency has to be reduced to accommodate this whole path to make it work.

But the another thing is a whole time violation that we know that at this point when a clock comes the data the rule is that whatever the data was there it should remain there you know it should remain there when I when a clock comes the data was there that should remain there

for some more time call whole time. So what we know that when a clock comes at the data here, the old data here will change to me after Tcq+Tnext state logic. So we take the minimum that means the data will change from TCQ+t next state logic, but there is a maximum value and the minimum value.

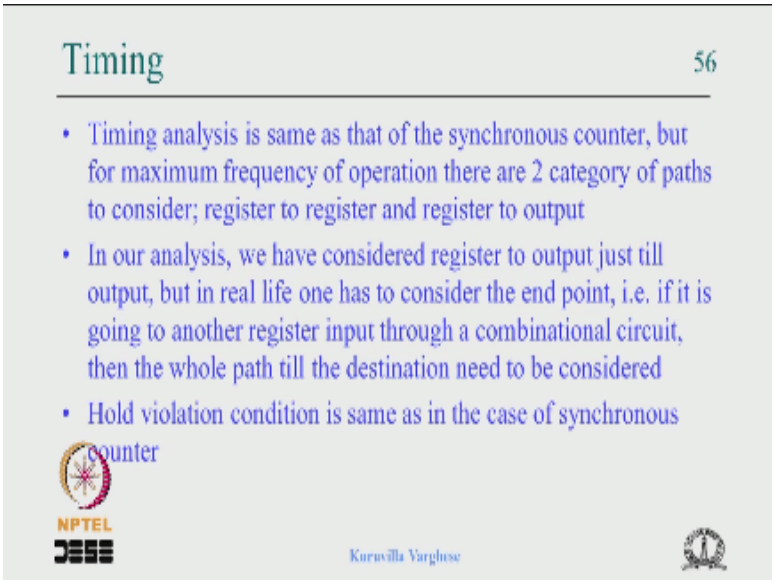
As far as whole time is concerned minimum is the critical kind of timings, so the TCQmin+T next state logic min should be greater than the whole time, the maximum whole time. So that is to violate the whole time and as I said that there is no clock period here, if it violates you have to increase the combination delay, why there is no clock period is that it refers to the same clock edge.

Because you now we are the data like a clock edge comes the data here has to remain there for some time, but the data changes at this point due to the same clock edge, so we are not like in the crock period analysis we are going from 1 clock edge to the next clock edge, but in the whole time violation we are consider why are considering the same clock edge like a clock comes the previous data, when the cock comes has to remain there for some time.

And we are working out because of the same clock edge the date hear you know how fast data here changes, it propagates hear, it propagates there and it changes there, that is why the clock period is not there. So if the whole time violation happens then it is you know nothing to clock frequency, it is you have to increase the combination, they are normally equal to like the pictures will show some 2 inverter back to back you put.

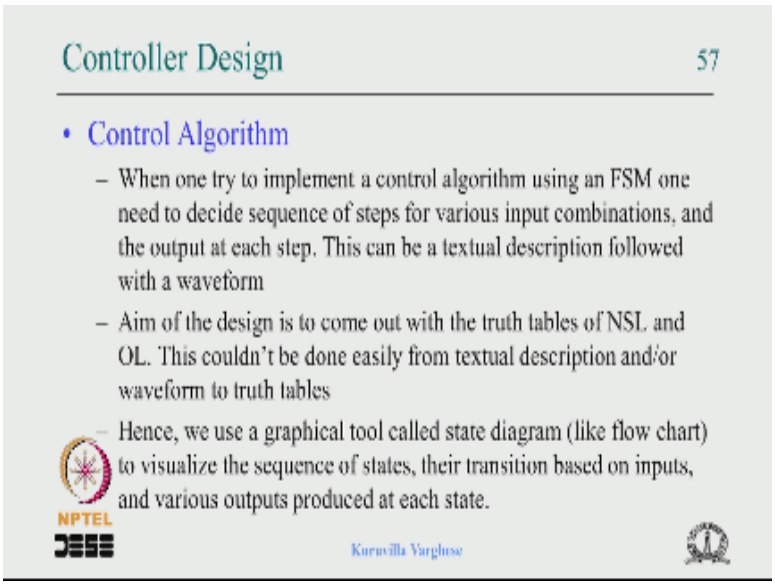
Then this problem is solved, so that is the timing analysis of the state machine. Almost exactly similar to the counter and the timing analysis of all sequential circuits looks same but I just wanted to reinforce that fact actually need not be considered separately, but then again for the learning that is useful to reinforce to consider this extra path which crop top and if you consider maybe the three blocks view that could have been maybe little easy like here you know you have one setup path here and other set up path there so whichever way you look at it is also same time.

(Refer Slide Time: 24:26)



So that is the timing analysis for the state machine and ok this is the part which have already discussed.

(Refer Slide Time: 24:31)



So let us look at the how to go about designing a controller and many times when you design a controller you have some description like if you have CPU what you have is a data path and you have some instructions and you know what is instruction does ok you marry me write it in text what the instruction does and now you have to convert that instruction into step in data path.

You know how the data path corporate in terms of the steps of data path that is the again maybe textual description but we are already bringing in the sequence, some steps or sequences and that is very useful we have designing a sequential circuits, so at the moment

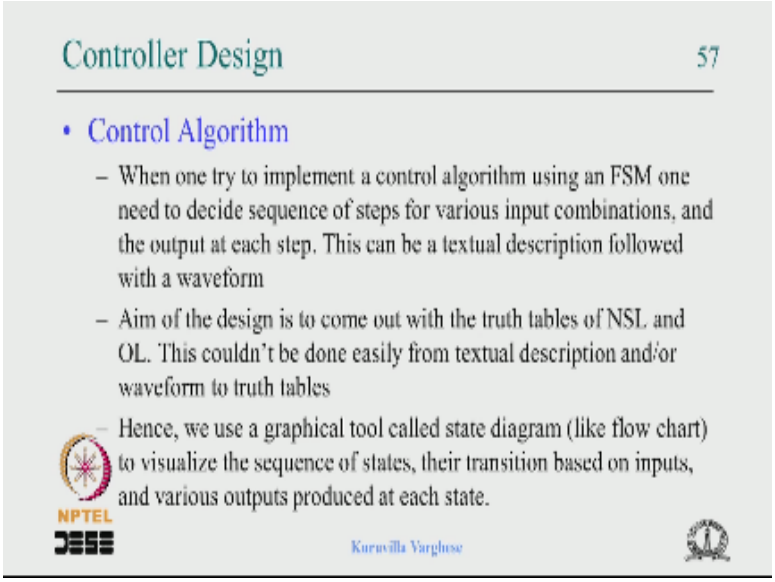
from an instruction to kind of steps or sequences is already here. We are coming near to the to the control algorithm.

Now the next step is to you right away for you know that is what we have done we have described like when we took the add instruction we have actually describe the sequence like moving something selecting something and so on then we put off the waveform then, so that is what is required now. So like when you design the control algorithms first step is always writing the various sequences, writing the waveform.

And ultimate aim is to design the next state logic and output logic, so for us that the third step is that you think convert this waveform, convert this sequences, it is stick, and it is almost straight forward because each sequence need a state like that some input comes then it goes to a sequence or some input come with go through two or three steps of 2 or 3 sequence in the case of add instruction.

The one instruction came and it just went through some three steps, but it may be different at like some input came it goes through two steps and it checked for some other input then it make a decision to go to a step A or step B and continuous and think sometime it comes back and so on. So that is designing a condolence, so let us look at the slide.

(Refer Slide Time: 27:12)



So basically you will have to decide the sequence of operation for various input combination and it is not enough kind of it cannot be done easily from my waveform to next state table or an output table or the truth table of truth tables of next state logic and output logic and many

times we are not clear that a lot of you know possibilities like in add instruction we could be a move A to Tier 1 and B to Tier 2.

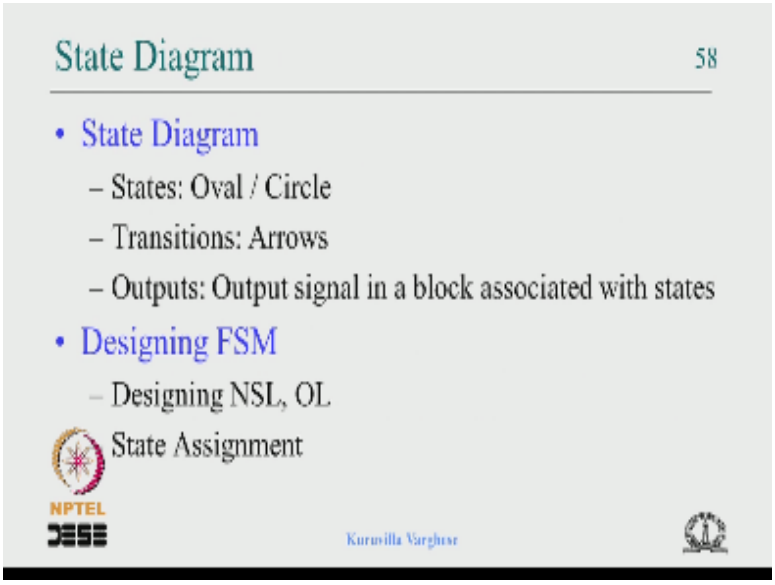
We could have done the opposite we could have move A to Tier 1 and B to Tier 2. So when you workout a sequence there are various other possible like there is no kind of it is not very clear at the beginning what all sequences what are the best minimal way of doing efficient way of doing the sequence also have to workout and we cannot work out in terms of the truth tables and so on ok.

So we needs to as soon as we are good at some kind of graphical tool to think and that is what happens in a software you know that there is there a flowchart or pseudo code which helps you in thinking like you have not you are trying you have a vague idea of an algorithm and your formalizing it in very clearly then graphic input tools will help ok, so we have a set of sequences set of waveforms.

And then we have to convert that in terms of the state of the controller, transition of the controller and depending on various input and output produced at each state, so all this can be done better than graphically. So as far as a state machine of controller is concerned there is a graphical tools similar to a flowchart and software and that is called state diagram. So state diagram helps you to visualise the control algorithm in terms of the state machine behaviour that what is basically had done.

So let us look at the that is what here is a graphical tool to basically to visualise it is for human being not necessary for kind of you know the languages of the software tools, but essentially will translate those kind of the diagram what is in the diagram to kind of equal and description for various languages.

(Refer Slide Time: 30:03)

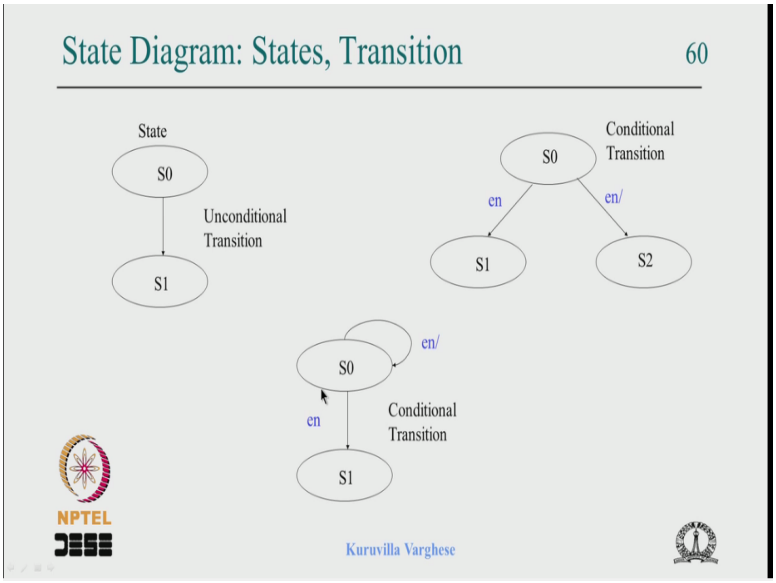


So that is a state diagram so basically what it means for the state diagram is that what we are trying to represent is we have to represent the states you have to represent inputs and the state transition like you say a particular input came in a particular state, then it transit to another state ok and in each state is produced some output, so that is what state diagram captures various states, various transitions which is the states are represented as an oval or circle.

Maybe like oval is better because the circle is very difficult to draw a proper circle all will be better because we can write something within and at least even a normal human being draw a circle you will end up with the oval, so maybe we can use all well and transition should be shown a arrows from this one oval to other oval and we have outputs various output real life there could be 5 to 20 outputs for state machine.

That is written you know some of connected to the state because the outputs are associated in each state. So basically that is the state diagram and he have to from the state diagram we have to drive this next state logic and output logic after doing the state diagram ok, that is the basic idea. So let us look at the state diagram, so once again keep this structure and review the state flip flops.

(Refer Slide Time: 31:56)



The next stage logic and output logic, 3 blocks you are the two blocks you and then let us go to the state diagram. So in the state diagram we have states transition and output. So let us look at the state and transition. So this is called unconditional transition that mean the controller present state was some state maybe S0 like if you are 3 flip flops it means maybe 000 0k 3 0s, unconditional transit means when the clock comes the state go to the next state like it transit from 0 0 0 to 0 0 1 or you go back to the diagram it mean this present said was 0000 and be decoded to produce output a clock comes respective of input the presence is decoder to the next state 001.

And the next clock comes that is loaded there that's meaning of that and you are in one state the controller is in one state up on the clock it transit to the next state, that is there is no condition it happens on the clock. There is no input condition, so that could like unconditional there are conditional transfer means the state machine present state is some state call S0.

And now there are input you know you look at here this is S0, there is a lot of input one input is called enable and if enabled is 0 the next it is decoded or something, if enable is one for the present state the next state is decoded or something else. So depending on the input value the next state changes and up on the clock that the presence it become the next state. So that is shown here like the present state S0.

And when the next clock comes if the input is enable is 0 then it goes to us to enable is 1, it goes to another state called S1 and outputs here and outputs here show different action

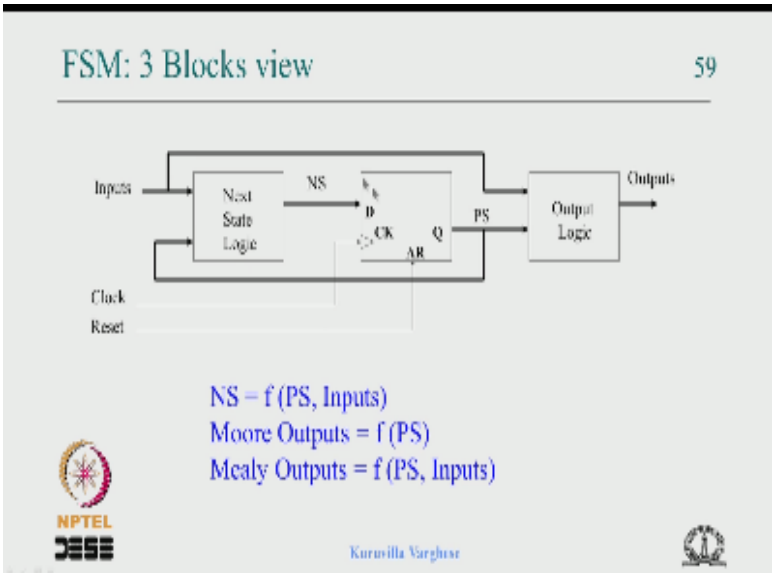
happened. So it is like in a fourth that your making a depending on some input you branch to some state and do something else totally different, maybe you can imagine in the case of a kind of CPU example, say this produces this is the state for fetching the instruction.

And depending on the instruction, type of instruction like say the move instruction come here I say that the add kind of computation instruction you come here and do something and so on ok. So the various you can connect it with the kind of applications in a kind of scenarios this can fit in ok. Now there is another very useful structure which is useful here at the present state is S0 and there is an input called enabled and it could be called anything.

And as long as that input is 0 it remain in that state and when that input like however many clock come it remains here and when that input becomes 1 it transfer to another state. So you can say that we are waiting for something happened ok. So like you can imagine that the controller is like something and you are waiting for that some signal from the circuit or you use you there is a data path which is doing some computation, you enable that that you know you are kind of started that computation.

You are waiting for it to get over it just a machine when it gets over it is continuing and doing something. Something like that very useful thing it wait for something and wait for something to happen and you remains in that is it. So it means that the state was something, some state and it is wait for some input go high from the data path, data path is doing something, is waiting for the state machine.

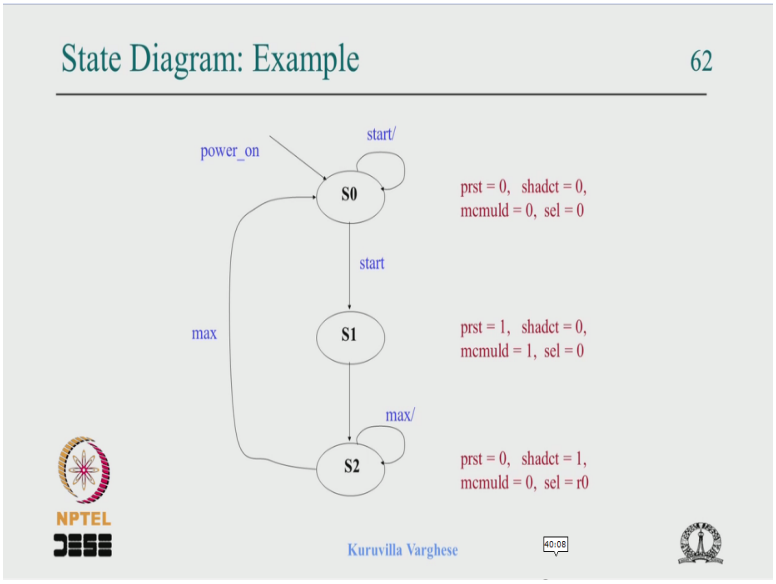
(Refer Slide Time: 36:16)



However many clock comes you know it can remain in that state for a day if the input does not come, so it just remain their and when that signal comes in transit was 1 and do something else, that is what is shown here, it is a conditional transaction, so many times it is called wait and go, like the you wait for something then go to the next state. So only these are the kind of 3 you know state transition possible.

That could be variation of these multiple transaction on multiple more than one signal you can imagine there are two signals controlling then there could be up to 4 transitions depending on maximum 4 transaction and so on. So let us look and output, now we have to handle in the keys of output both Moore and mealy output.

(Refer Slide Time: 37:09)



So like this is the Moore output I have shown here in this state say there is a signal called V bar which is 1 and latch 0 ok. So that is what is shown here and this is associated with this state, it essentially means that the S0 is Decoder to produce read bar 1 and latch 0 and in this state the read bar is 0 it is transmitted like change from 1 to 0 and latch may be 1 and maybe in the next state and the latch is 0 then you get pulse on latch.

So 1 clock period duration, in case of Mealy output then you have the it is latched has a conditional output, so I hear you see the FSM is in the state S0, it is waiting for this enable signal to happen and when enable signal and as long as it reminds in this the latch is kind of 0 because latch is one when enabled is 1, that is that is condition here. So when enable comes it does two things.

It transit to the next state upon the next clock and in this state itself the latch becomes enable becomes 1 the latch becomes 1. So that happens in this state before the state changes and this means in the next clock enablers become one, in the next clock the transition happens and you know that it has to be set up before for the transition to happen because we have this, is the transaction has happened then that enable has come sometime before the setup time and the logic delay then only the transaction can happen.

So the enable in this case has come properly much before, so when does controller is in this state and the enable comes before during that duration the latch will be 1. So that meaning of it. We will see the timing of the Mealy output later, but that is the, that is how the state machine looks like ok.

So let us take and example state diagram, very simple state diagram and this is not something a kind of arbitrary state diagram of a controller this is the state diagram of a multiplier which we will take a case study later, but I just want to kind of highlight how a real life in diagram looks like. So it is very simple state diagram. So this we have 3 states and you see S0 and this shows the power on.

You have to bring it to a starting state because we have to start in some known seat so we will see how to bring a controller to a power on state and then it is waiting for a start signal. So here this is output, there are 4 output call Preset, kind of shift signal, load signal and select signal ok and upon the start so it shows that it is waiting for a start signal from outside upon the start, so this is the kind of power on initialization everything is made kind of inactive.

Upon the start signal it comes to a state called initialization estate, it initialization as the data path ok. So reset is made one, some loading happens in the data path, then it goes to a computing state where number of iteration happens and it is waiting for that preparation then we see how that happens it is looking for the computation to stop and that is called a Mac signal.

As long as the max signal is low the computation happens and this is the kind of ship signal is one, for whatever reason we will see that later and it is selecting some path if in the data path and then the signal comes from the data path it goes back to the starting state waiting for the next start signal and so this is a kind of real life state diagram of a controller of multiplier.

And you see that the inputs decide the state transition and there are 3 states and from each state there are kind of output associated most of it is Moore output but you see this straight line is a Mealy output because select line is a function of some signal R0 which not from here, we know that it need not be the case that some of the input coming to the state machine need to produce the Mealy output you know is not required.

It can be something else, but which is related to the data path your controlling naturally that has to happen. So the moment you do this the state diagram then we can write the next state logic and output logic, that is the most important thing. So like look here and before going that we have to assign some state kind of state assignment that need to be done.

So here if you do binary encoding for three stage we need 2 flip flops ok. So let us assume the states are like 2 flip flop Q1 and Q0 let us assume a sequential state assignment saying that this is 0, 0, 0,1 and 1, 0. So how does the next state logic looks like, you know if it start like that at the power on next stage is the present stage 00 and input signals are there, when start signal is 0 the next is 000 and remains here.

When the start signal is 1 then the next state is nothing but 01. So looking at the state diagram you can write the next table at the moment you have the state diagram then you can write the next state, same thing is true of the output table the truth table of the output logic or output table like you see in the state S0 the P-reset is 0, shift 10 load is 0, select is 0.

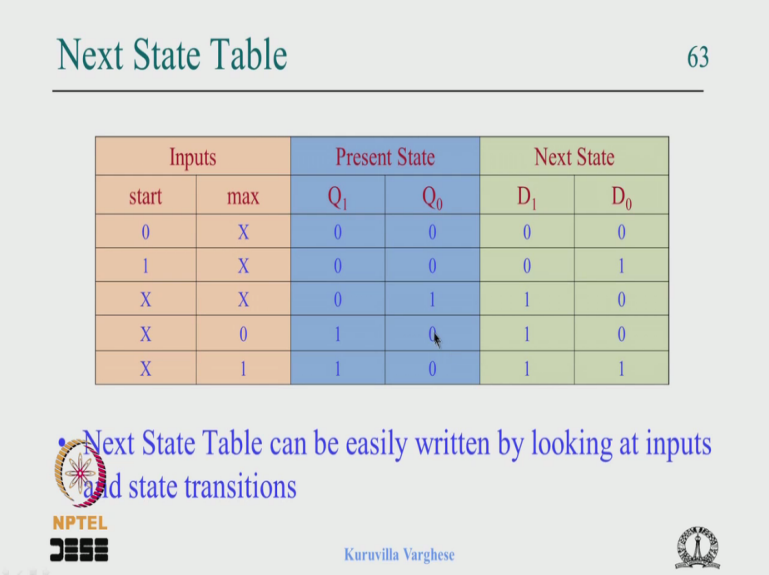
But when it comes in the state S1 or 01 reset is one, load is one restore say ok. So that is that can be easily return in a table and one thing I should mention that many a time some text books will show that like for kind of brevity or for a concise representation it will show the signals that change, I means do not write everything you have 10 outputs, do not write and outputs everywhere see here if you see reset and load only changes this remain so do not write it.

But this as far as a figure is concerned it will be charged me but when you write development next a table or output table or write a code to do this you can miss that point on write you will get in doing a lot of trouble. So I do not suggest that you do such a thing ok. There are two

other ways to kind of circumvent that problem by initialising some default values and so on in a in a coding but is not a very good idea.

So you should it is better you write explicitly what are the outputs ok. I am now only I am warning kind of various pitfalls people are the mistakes people make a make sure when I know when you are designing other state machine.

(Refer Slide Time: 45:42)



So let us look at the both tables next state table and output table so that is what is captured here like this is the present state Q1 and Q0 we have two inputs start a Max only two inputs are there, this will start and max. So that is what is shown here and we are decoding the next state that is it if function of the next table. So initially are the power on it is in 00 the present state is 00 when the start is 0 the max is do not care.

We do not care about Max then the next state is 0-0 itself and when start becomes 1 max do not care 00 transit to 0, 1 and if you see the next transition when you are in as 1,0,1 there is no condition and with 10, so when you are in 01 respective of the condition the transaction is me to 10 and again when Max is in your own 1,0 maths is low or inactive then domains there waiting for the Max 2 become 1.

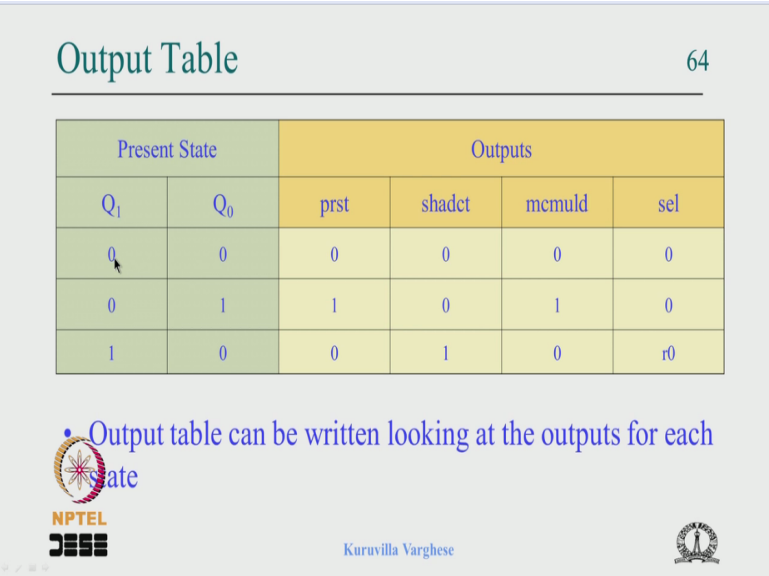
And when Max is one it transit 2, 1, 0 is transit 1, 1. So you basically get 2 equations for D1 and D0 normally we take we form the equation in terms of the Q1, Q0 there, in word minimise it and all that not required this can be easily coded and the tool, so this tool will do the optimisation and kind of the create the necessary circuit to implement the next state logic.

Mind you that this state diagram what are the state and transition part and the next state table is equivalent, you know we have returned the table from the state diagram only. So many times a design we need not write this table, we describe the table in terms of hardware description language looking at the other state diagram. So the moment you design the state diagram the controller design is over we describe this next that is all in in hardware description language then the design is done.

So let us look at now the output logic and I say as I said is not good too kind of an awesome text book will show you that you know even write output here various output and you say for 00 what are the output but then it is very crazy because this being a next state logic table 0, 0 is repeated twice. So you are forced to write the output repeated twice and we have know it to know whether the output is a function of the present state or input.

And there is confusion and sometime people write the input condition on top as a kind of row label, this is a column label and the resulting state and all that. In that case cannot be combined at all but I suggest that you do not complicate is a simple method to represent it separately represent conveniently needs for you to study in nowadays in real life there is no need to do right next state table or an output table.

(Refer Slide Time: 49:06)



So this is output table like present state is the function of the present state, present state 0,0 then you know that everything is 0, all the outputs are 0, 01 does reset and load is 1 and 1, 0 this shift signal is 1 and select is a R0 is not shown here but maybe you can show the input as

also in the table does not matter but then that is that is the way it is again. This is exactly equivalent to the state diagram.

Because it specify for inside the output the kind of the output logic is kind of identical to the state diagram it is enough. We write the hardware description looking at the state diagram, moment you have done everything is done and captured as described in hardware description language from the like we can describe the logic and output logic looking at the state diagram.

That it is about the state diagram and the next state table and output table. So like before going for the time we have like we will start with the case study at least I will mention the scenario and do the design in the next class. So just now we have seen we have done the timing analysis of the state machine basically the register to register path and register to output path possibly extent to another register path, whole time violation.

Then we have seen that you go from the text description waveform by coming out with a sequence of steps and ultimately translate the sequence of steps into the state of the state machine and various transition and various output it produces and too many times as I said control algorithms are not easy to kind of ah you know design because there are various possibilities we need to work through, it is not that straight forward you get the algorithm.

Maybe there are like when you walk with the CPU case you will find there are common path like you might have come out with the state diagram for various instructions, when you find that there are instructions for which the number of stage are equivalent, so we can combine them together and all that. So it needs lot of a kind of work on a piece of paper and the graphical tool is coming in so the so we have graphical state diagram which represent the state which represent the state transition depending on the input and which also shows various output h.

So we have some kind of the state of oval I will suppression the transition we also seen unconditional transition where up on the clock a present state goes to the next state and conditional translation depending on input it make some branching, 2 or more branching and there are also we can go structure which way for a particular signal on a particular group of signal to change the state.

Then goes to the next state and Mealy and Moore output are normally outputs are connected to a state and the Mealy output is written as a function of the input. That input may not be the part of the state transition that you should keep in mind, it could be some separate input coming from the data path and I also said that I do not use shortcut when you describe the output do not write just output with changes.

So that there is no confusion and we have seen in an example of state diagram and we have tried to work out the next state table and output table from the state diagram what turns out is that the next state table we write, all the information is in the in the state diagram, output table we write all the information is in the state diagram, for the moment one somebody draws a state diagram the control algorithm is fixed.

And the everything required for the state table, next state logic and output logic is there. So it is enough in our methodology we code this in a hardware description language the thing is done. So I think kind of you get a picture of a kind of the controller what is its structure, what is how it is designed, what are the tools we use graphical tourist designed the controller.

And now we will illustrate it is not that unless we do some of real life scenario this may not become very clear, so I will take a simple example where the concentration is on the controller not the data path. So because if you put a data path then we have to design the data path. So I have taken some make one example where there is not much of a data path, but then the that there is a controller which is a very it is not that I took up something, very simple.

Its whatever I describe is a very useful thing in the next like at the case study. So our idea is that we go from the word description to all these procedure all the way up to the next state logic and output logic. So you can relate the oldest up know that that in the in the present scenario that we are not doing everything by hand, most with is done by the tool but there is a part where the designers involvement is quite high.

So I will illustrate every step involved in the design whatever the designer does and even the tool does ok. So that you get a clarity understanding that is what is the lecture is this course is about I want to bring the concept very clear in your mind, what the designer does, what the

tool does without much relation to particular tool. As we go along I will show all that thing in some kind of design tool definitely that illustrated.

But this case study this is a kind of simple case study to start with we will discuss in the next lecture. So do not take it very lightly this current topics. Please go back look at it analyse it and learn it well, it should be clear in the mind you know it is not that you should work out in the mind and you should be able to visualise all these in the mind, then because when you design complex thing will be good if you can kind of design this things in the mind. So I wind up lectures here. Thank you and wish you all the best.