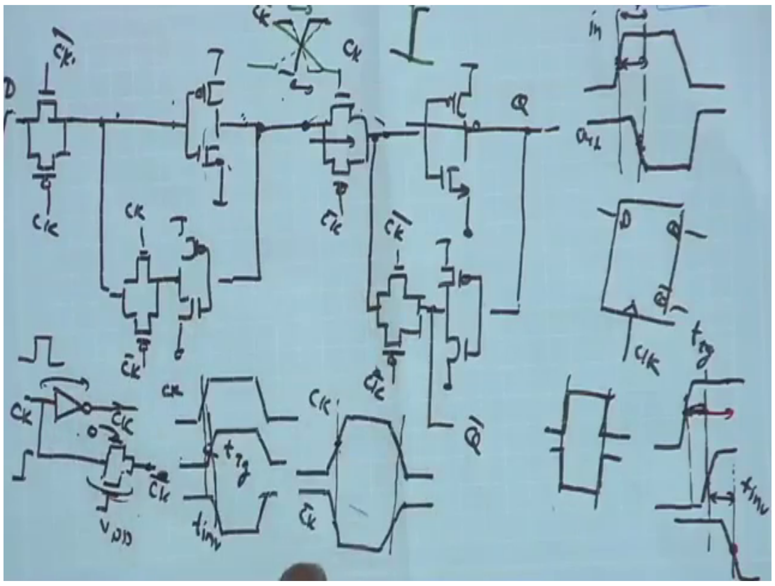
Analog Circuits and Systems through SPICE Simulation

Prof. Mrigank Sharad
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture – 52

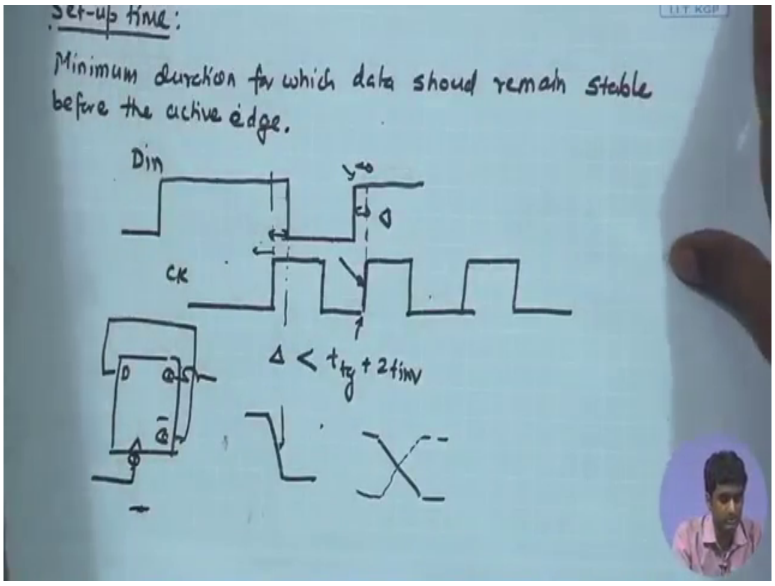
Let us continue with the discussion from where we left here.

(Refer Slide Time: 00:22)



Any question regarding these 3 definitions? So, in our case of course, the important criteria is the delay 3 c 2 Q delay because we are looking at the ripple counters c 3 2 Q delay can be important. And if you look at the setup time; however, if we look at the counter operation are we concerned about the setup time in our ripple counter. So, if I see where are we applying this.

(Refer Slide Time: 00:45)


Set-up time:

Minimum duration for which data shoud remain stable before the active edge.

So, you have the toggle flip flop being constructed D Q and Q bar. And we know that Q bar is connected to the D and then you have the clock. So, each of the element in my design is having the Q bar connected to the D and therefore, before the positive edge of in our case we have used a negative edge triggered just for the up counting operation we have used the negative edge trigger.

So, before the negative edge comes the Q bar is anyways stable right. So, Q bar is always stable because much before the negative way of the clock Q bar was equal to the negative of Q. Therefore, Q bar was always stable. I do not need to worry about the sudden change in Q bar just before the down going edge of the flip flop in our case because Q bar sensitive D. So, the D is automatically only the negative or inversion of the Q which was stable all throughout the period last period. Therefore, the setup time will not be an important concern in our case; however, clock to Q delay will be and concerned will be a concern because we a concerned with the total delay of the ripple propagation from the input to output. What about the whole time is it concern? If you have bad transition time for the clock and clock bar, which are going into this flip flop of course, I have shown one clock, but ultimately you have to produce both the clocks lock and lock bar which are going to control the pgs in this flip flop and you are connecting the Q bar to the D.

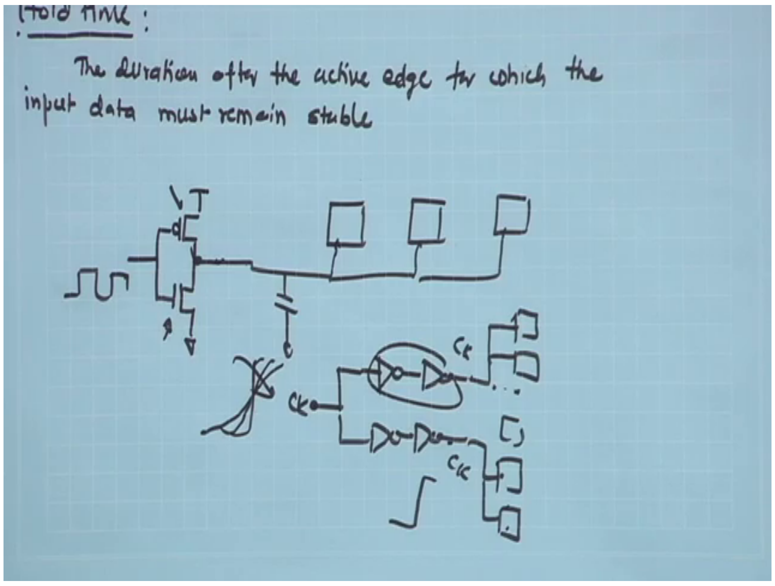
So, basically what you have done is this Q bar in this circuit is connected to the D. And if you have the clocks which are having clocks which are having say transition my Q is supposed to change at the positive edge of the clock. So, whenever there is a positive edge of the clock Q is definitely going to toggle that is the operation the toggle flip flop. So, at every positive edge Q is going to toggle. And therefore, at the positive edge of the clock we expect the transition in Q it is going from high to low or low to high. And therefore, there can be when you have a overlap between the rising edge of the clock and there is a significant overlap between the 2 phases. You can have a transparent path and rather than Q bar getting reliably lashed to the D you can have intermediate undefined state. Because you are connecting Q bar over here in D and all these latches are on for a given duration. So, it is not necessary that the Q bar will be reliably copied back to here because the polarity from this Q bar to this point may not fall to 180 degree at that same instant because all these latch are getting on together. And as a result I can have the wrong transition over here and it can lead to aarons phase in the counter output.

So, the whole time will become important and the rising falling edge rise time for time edge and the overlap in the clock and clock bar will become important in our application, we would like to make sure that the rise fall time is not that bad that when you are connecting this Q bar directly to the D, and for a short duration these are remaining on together you are having a intermediate state and the Q bar is not copied properly to the final Q.

So, that is the only concern that we take care of So, you should make sure that the rising falling edge of the clock going to this transmission give there not to very bad, because otherwise it can lead to strong latching. Because if you are trying to say Q bar over here you have in this path you do not have any inversion inverter therefore, logically if this rhythm is connected over here to the D. And both of them are having overlap both of them are getting on at the same time. Then basically what you are trying to do is Q bar and Q you are trying to connect together the moment you have overlap. And you are putting the Q bar over here from here to here one inversion here to here another inversion. So, you are trying to basically set Q bar equal to Q in that transition duration which is not wanted we can have a undesired output you can have wrong transition over here.

So, that is something to be careful about and that can be ensured if you are having sufficient rising falling sufficiently sharp edges of the clock. And sufficiently small rise time and fall time. How does the rise time fall time can is controlled? So, just a brief discussion on that So that you know we can end the discussion quickly.

(Refer Slide Time: 05:31)


Hold time:

The duration after the active edge for which the input data must remain stable

So Rise time and fall time of the clock are going to be controlled by the buffers or inverters which are driving the clock ultimately, as I said the clock will be coming from some source which is having the signal 0 to V DD going in and you may have a single inverter driving multiple transmission gates.

So, a single inverter may be responsible for driving the clocks of multiple flip flops or multiple gates. You can have a single clock input going to multiple blocks. In our case the first clock anyway come with the first flip flop and output of the first flip flop goes to the next flip flop and so on, but in case the output of a single source of the clock is driving many different blocks. Then the load capacitance faced by this inverter or the driving circuitry can be pretty large, and as a result the rise time for time can degrade. The moment you have more load connected to a particular driving gate are digital gate the rise time fall time can degrade. So, this is larger and larger number of blocks connected to the single driver or single inverter you can have degrading rise time and fall time. Therefore, this condition should be avoided in case a single gate is supposed to drive large number of other digital blocks, you can try to make the size of the NMOS and PMOS larger

So, that correspondingly the current pumped in by this PMOS and NMOS also becomes larger and as a result the rise time whole time is maintained. So, this is something to be careful about whenever you are using inverters logic gates to drive the next pages especially in case of clock is the rise time for time is too degraded you would like to make the driving source maybe the inverters with larger dimension. Or you would like to use to inverter rather than using only one inverter this is the main clock I can use to inverter this is the ck I can produce say, I can use 2 buffers and produce ck over here this is the buffer right 2 inverters together means buffer you have no inversion in this path.

So, in that case rather than the one single clock driving to many blocks, I can have effectively 2 clock sources. So, have divided the single source into 2 sources and therefore, the effective load capacitance will be half. So, digital signal or digital clock can always be buffered you can have bifurcation and then you can have 2 sources driving the half the loads each another result their fan out their total load capacitances will reduce and there is the important definition of fan out that is that comes in digital circuitry, which deals with how many similar gates are drawn by a single proceeding gate.

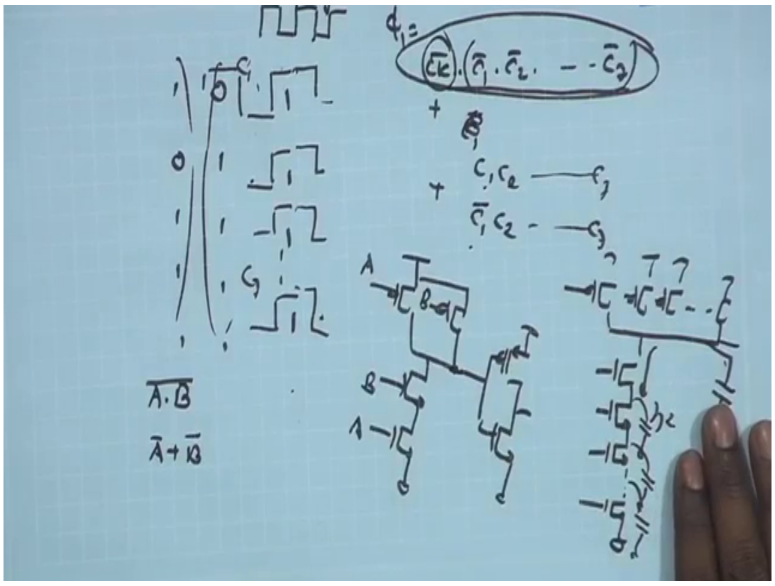
So, in terms of analog circuit we talked about parasitic capacitances and load etcetera in terms of digital circuitry because we are dealing with individual gates we talk about fan out. So, in this case if you have more number of gates being driven by a single gate the fan out of that gate is large. So, here if you have 3 such gates 3 logic gates will be and gate or gate or not gate being driven by the previous gate; that means, the fan out of this gate is 3. If you have n different gates given driven by a single gate that fan out of this gate is n times. So, larger is the fan out you need to keep the dimension of these transistors also larger. So, that it can supply appropriate charging and discharging current and the transition can be faster.

So, this is another important point whenever you are looking at digital circuitry the their transistor level implementation you should take care of this fan outs. That a single gate driving many different gate it will lead to degradation of rise time fall time one option is you increase the size of the driving it other option is you split the signal you use buffers like this. So, that the load or the fan out of the individual buffers and the output is lower here for example, I have split it into 2 halves each of them tethering to half the load.

So, this can help you in retaining the overall shape of the overall sharpness of the rise and fall time of the data in the clock. Any question? We proceed? So, these are important considerations by looking at transistor of implementation. So, just like for the analog circuit and the switches we have seen some consideration of regarding the (Refer Time: 09:53) switches when we are talking about flip flop it is important to be aware of the nonideality of the flip flop. And be aware of the transistor level implementation on those flip flops.

Another small topic related to the digital logic implementation is the logic we discussed for generating the phi 1. If you remember we have the some small logic being used to implement phi 1.

(Refer Slide Time: 10:28)



So, we looked at the counting operation and whenever all the Levels are turning high. So, you have say c 1 to c 7 all of them at some instants they are turning high and then finally, going low. And we used the low phase of the clock and we said that it is negative edge trigger and we said that, we are going to and this clock bar and c 1 bar c 2 bar c 7 bar to generate the phi 1 pulse. So, this was the phi 1 pulse was given by ck bar dot c 1 bar dot c 2 bar dot c 7 bar that is how we generated the 5 pulse. And the other the question was if you want to have wider pi one pulse and only thing is you would have to use more such logic.

So, for example, you want to have phi 1 pulses being little wider we consider that these reason why you would like to have 7 pulse little wider in the last discussion the feedback of the OPAM etcetera, can take more time. So, in case you are not able to speed up the feedback of the unity gain offset cancellation loop or the sampling circuitry you may have to apply a longer phi 1. That is what we discussed and for that only thing you need to do is apply more logic that it the for more counts like here I am talking about 1 1 1 1 all ones before that you will have 0 1 1 1 1 before that you may have you know one 0 1 1 1 1 and so on.

So, I will apply the logic responding to this combination this combination and set them all to 0. So, whenever this particular phases are coming for the counter I would like to keep for this entire duration, I would like to keep the phi 1 high, I will just use that additional logic. So, basically it will have another or gate another or gate. So, you get the first combination over here I may not even use ck in that case I will just use the count c 1 c 2 c 7 bar. Then I will use say you know c 1 bar or basically c 1 or for example, if I have to you know keep the phi 1 on for a longer duration and I am looking at this phase. So, just before this you have the phase all 1.

So, I can keep this c 1 c 2 up to cn c 7. So, this will be the phase just preceding this one before that you will have the phase c 1 bar you know c to up to c 7 proceeding just this 1. So, suppose for this 3 phases I would like to keep the phi 1 high I will just take the or of this. And this therefore, you calculate or you compute this particular logic using and function another and function to calculate or to compute this particular combination another. And function to compute the third one and you or them together and therefore, the phi 1 will be remaining on for that entire duration.

So, you have basically the and gates taking care of this min terms are called min terms combination of this and functions, and then they are all together. So, basically bunch of and gates forward by or gates.

Now, there is a question regarding implementation of these and gates and or gates at transistor level or do we implement this the So, basic structure we know for the for example, if you are trying to implement and nand gate from which we can implement an and gate we know what is the what is a nand gate look like. So, it is something like you are having 2 input nand gate where you have the input A and B coming here and A and B coming here. And we know that when both A and B are high then the output will be low and then you want to get a and gate out of this. So, put an inverter. So, this is a and gate.

Now, if you want to if you want to implement this function over here can we go on adding. So, if I go for 3 input nand gate or 3 input and gate I just need to add another c over here third transistor coming over here and another third transistor in the parallel. How does that parallel combination come? So, basically by the de morgans law if you are having the a dot B this output is supposed to go down when A and B is high. So, this is implementing basically a dot B bar. So, this NMOS combination is implementing A dot B bar because when A and B both are high the output will be driven low and we know de morgans law, that this can be represented as A bar plus B bar a dot B bar can be representing as A bar plus B bar.

So, this PMOS circuitry over here should also support the same logic condition. So, when a dot B is high. Then the PMOS should be driving it height. Then we can see that when a here if you have a low and B low in both conditions output will be driven high and that is basically nothing is, but this function right. So, this we can implement this A bar plus B bar by connecting A and B transistor in parallel using PMOS because here when a goes low then this pulls the output high B goes low then this pulls output high. So, we take the de morgans we apply the de morgans law on the pull down logic and arrive at the transistor combination needed for the PMOS.

So, here we are seeing A bar plus B bar that is obtained by connecting these 2 in parallel. And they implement the same function (Refer Time: 15:58) when this is trying to pull it down in the opposite phase this will be trying to pull it up. So, both of them are satisfying the condition.

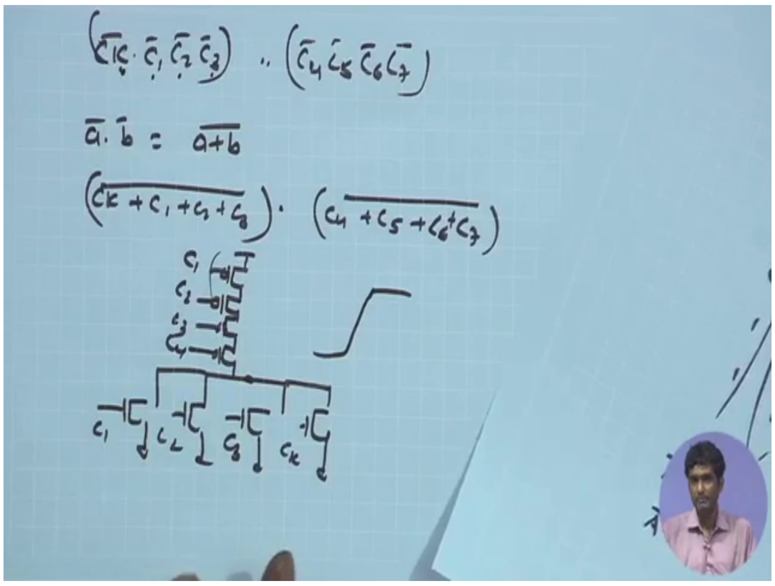
Now, if we just go on adding these gates to implement larger logic functions you will have the stack of n number of transistor you will have you know 7 transistor 8 transistors coming in, and you will have a huge chain of transistors over here. So, if I want to implement the and of these all these functions together I will have to use such as pack c 1 c 2 c 3 c 4 c 7 coming till this point and also the corresponding c 1 c 2 c 3 c 4 coming here. And then ideally of course, I can do this and in digital circuit we know that we do not need to worry about the voltage headroom just like we do in analog circuitry. So, these are just try out reason switches and ultimately the voltage drop across them can reach to zero.

So, I do not worry. So, much about the triode the voltage head room across the switches in digital operation. But what I do need to worry about is the transition delay. So, if you are stacking too many transistors in this fashion, each of them are going to add to the overall transition building of the discharge path. And that increases at n square if you have n such transistor that overall transition delay of this logic increases as n square. Because effectively you are having parasitic capacitances that each point which is increasing with n total capacitance charged, the worst case and also we have the total r in the series connection which is also increasing with n. Therefore, the total delay can be shown to be proportional to n square.

So, if you stack more and more transistor to implement such logic it can increase the delay steeply. Therefore, generally people do not go beyond 4 input. So, maximum number of input that we can see libraries standard cell for input better is maybe 3 inputs and so on. So, you know beyond for input it can become it can degrade delay pretty significantly. So, in our cases we are having say 7 or 8 input in our original case we started with this particular combination ck bar c 1 c 2 c 3 c 7. So, we can split it into 2 halves 4 4 and then implement the ck bar c 1 bar c 2 bar up to c 3 bar and then another one and then and them together that can be one solution.

So, I can rather than implementing the entire function using a single gate I would split them into 2 halves and then combine the result. So, basically I am going to look at ck bar

(Refer Slide Time: 18:34).



$(\bar{c_k} \cdot \bar{c_1} \bar{c_2} \bar{c_3}) \cdot (\bar{c_4} \bar{c_5} \bar{c_6} \bar{c_7})$
$\bar{a} \cdot \bar{b} = \overline{a+b}$
$(\overline{c_k + c_1 + c_2 + c_3}) \cdot (\overline{c_4 + c_5 + c_6 + c_7})$
c_1
c_2
c_3
c_4
c_1
c_2
c_3
c_k

c 1 bar c 2 bar c 3 bar. And another one c 4 bar c 5 bar c 6 bar c 7 bar implement them individually using gates and then take their and. And if I look at this is if I look at A bar B bar you know that this is nothing is what a plus B bar. So, I can also represent I can extend it to the larger number of inputs also therefore, this can also be implemented as nor gate. So, I can just have ck plus c 1 plus c 2 plus c 3 nor given as this combination. And then it is anded with c 4 plus c 5 plus c 6 plus c 7 and then again you have the nor of these two. So, basically you need to implement for input nor gate. And then and the result together. And as a result means nor gate can be implemented using once again similar scheme only thing is the combination will just oppose it you have the NMOS in parallel the for NMOS coming in parallel c 1 c 2 c 3 ck and then you have the PMOS coming in series because if I take the this is going to be low when the any one of these is high. And then the negation of this the de morgans we know the negation of this is going to be this ck bar c 1 bar c 2 bar c 3 bar.

So, that comes in series, c 1 c 2 c 3 c 4. So, this is a 4 input nor gate I can take 2 such gates and apply I were these 2 and then combine them using and gate. That can be possible implementation and one important point to be noted is in general, you can have the PMOS mobility the mobility of the PMOS invention meter can be half the mobility of NMOS. And as a result the speed of the PMOS transistor can be half the speed of the NMOS transistor for the same dimension. So, therefore, if you stack PMOS transistors it leads to slower speed rather than if you track NMOS transistor, if you stacking NMOS transistor the speed for 4 stack NMOS transistor can be better. Significantly better than the speed of 4 stack PMOS transistor because mobility of the PMOS is slightly lower or rather 2 times lower in manatee nanometer. That difference has vanished for lower technology for lower technology note that difference is very minimal, but for older technologies one tonometer is still at least 2 to 2.5 ratio you will get.

So, generally for high speed logic we may prefer nand gate implementation, but in our case that is not an important concern because we are not that where logic speed is not very high. As a result we can go for nor 8 implementation also. So, on we just should just become aware of the logic design if you are supposed to implement any logic gate using transistors you should know how to split it and how to implement it using smaller gates. So, that the delay is within limit and you are having right kind of transitions, if your along with the delay of course, another point is rise and fall times you stack more transistors because of the large rc time constant the rise time fall time will also be degraded. So, this issues should be kept in mind while implementing transistor level logic any question before we end this discussion.

So, we have basically covered the building blocks of our adc in sufficient detail starting from the ramping circuitry comparators there are non idealities sampling circuitry along with the comparator design it is specifications it is non idealities including offset cancellation. And finally, we also came into the digital controller part where although the control is simple, but we dealt with the if the constituent blocks constituent elements at block level overall functionality of the control operation how the 5 and 5 2 pulses are getting generated. And finally, we get went into the transistor level implementation of all those digital blocks look into the important definitions and issues at transistor level implementation of those digital components.

So, this basically completes the overall controller module that we have for the a to D converter. Any question before we end this discussion?

Student: Sir if a speed of the mos is lower then we other output synchronize?

Synchronize means clock with respect to clock as you will have we know little bit more delays, if you are applying the equal to a PMOS or applying input to a gate where we have more PMOS stack then delay will be slightly longer. So, as compared to the clock edge the delay may be you know slightly more that is the only thing.

Student: (Refer Time: 23:17) sir PMOS output keeps some output and PMOS output.

When the NMOS is off, then the if the lower branch is off then only the upper branch will be turning on and will be charging the capacitor. So, they are not on at the same time. So, there is no fight they are complimentary when this off then only the other goes on.

Student: Sir (Refer Time: 23:38) divided into 2 part. So, why can not we simply derive within 2 parts without (Refer Time: 23:43) buffer.

So, miss with if you do not add the buffers and how are you going to, if you do not have this inverter then this load capacitance remain same.

Student: (Refer Time: 23:56).

That is the main purpose right to dividing the load caption is making it smaller. So, if I if this was connected to you know 8 logic gates then the fan out of this was 8. Now I inverted infrared some buffers and then this buffer is driving for this buffer is having 4. So, for my fan out of each of these 4. Any other question? So, we can end the discussion and resume with a new topic.