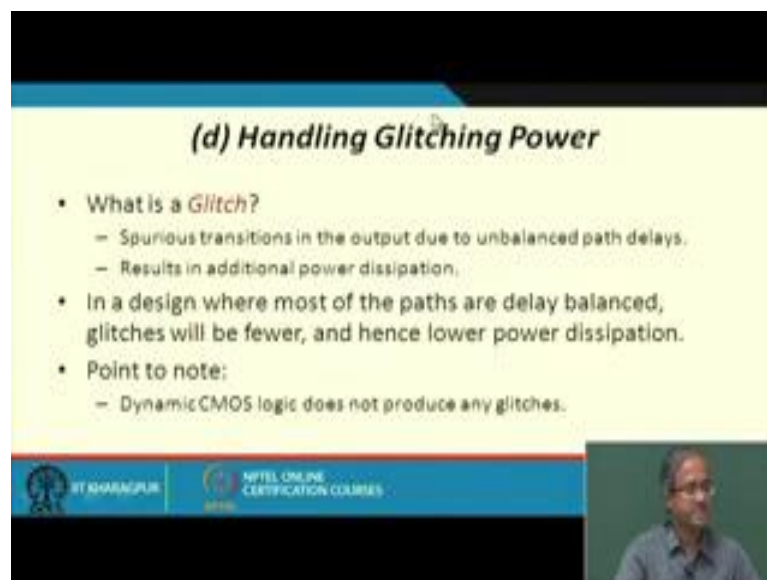


**VLSI Physical Design**  
**Prof. Indranil Sengupta**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture - 61**  
**Gate Level Design for Low Power (Part 2)**

So, in this lecture we shall be continuing with what we were discussing during the last lecture. We will be looking at some of the techniques for low power at the level of gates or at the level of functional blocks RTL level. So, we continue with our discussion.

(Refer Slide Time: 00:40)



**(d) Handling Glitching Power**

- What is a *Glitch*?
  - Spurious transitions in the output due to unbalanced path delays.
  - Results in additional power dissipation.
- In a design where most of the paths are delay balanced, glitches will be fewer, and hence lower power dissipation.
- Point to note:
  - Dynamic CMOS logic does not produce any glitches.

The slide also features logos for IIT Kharagpur and NPTEL Online Certification Courses at the bottom, and a small video inset of the professor in the bottom right corner.

So, the first method that we look at this concerns Glitch. Talking about Glitch, see Glitch is your hazards in a circuit. These refer to unwanted signal transitions because of the delays in the gates the circuit elements what might so happen is that the output of a circuit is supposed to make a clean transition, let us say 0 to 1, but because of these unequal delays the signals or the changes take unequal times to reach the output. So, there can be momentarily some transitions before the output settles down. These are called Glitches or hazards static hazard dynamic hazard.

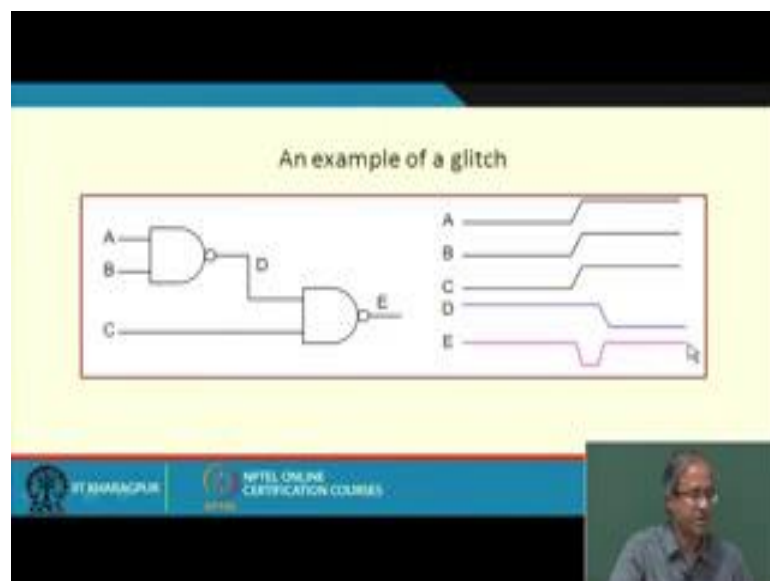
Now, see if your circuit has been designed in such a way that there are Glitches like this, what does that mean. Glitch means the output node is changing unnecessarily a few times, which means charging and discharging of the capacitor is taking place

unnecessarily that many times which means extra power consumption. So, if you can avoid Glitch, this kind of power dissipation will also go down.

So, here we are specifically talking about this kind of Glitching power. So, as I had said a Glitch which sometimes is also known as hazard. These refer to spurious transitions; that means, transitions which are functionally not expected to happen, in the output due to unbalanced path delays. We shall be showing in some examples. Because of this, there will be additional transitions, additional charging and discharging taking place on the load capacitor. So, this means additional power dissipation.

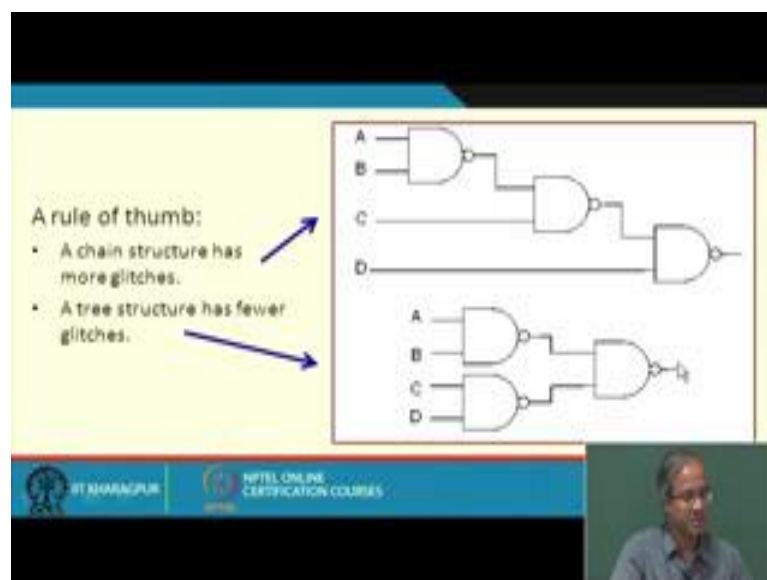
So, for designs where the paths and the delays are balanced, which means from the inputs the primary inputs to the outputs, the delay is through all the paths are approximately the same, these kind of Glitches or hazards are unlikely to happen. Only for those cases where there is some kind of unbalanced nature in the circuit, there you can have this kind of things. And just to notice that dynamic cmos logic does not produce any Glitches because here, the way the circuit works pre charge and compute state this, kind of Glitches will not happen there. Because the output is definitely pre charged to a level high, then during compute state it will only it can go to 0. So, it cannot go to 0 again to high again to low that is not possible. So, this is a point to note. So, dynamic cmos circuits are good from this point of view.

(Refer Slide Time: 03:49)



They will have 0 Glitching power. Let us take an example. This is a very simple circuit, a 2 level circuit consisting of 2 nand gates. Let us say the inputs are changing all of them from 0 to 1. So, we assume that there is a small delay in the gates. So, D they are changing 0 to 1. So, d will change from 1 to 0. So, after small delay D is changing from 1 to 0. Look at the output E. So, when D is changing this C is already high. So, there is a period where both the inputs are high. And there is a period where one is high other is low. So, because of this there will be a temporary period where the output will go to 0 before again settling back to 1. So, this is a Glitch which is also called as static hazard; that means, the output was supposed to remain at 1, but because of delay in balance, it is temporarily going down and again going up, which means one discharging and one charging.

(Refer Slide Time: 05:03)

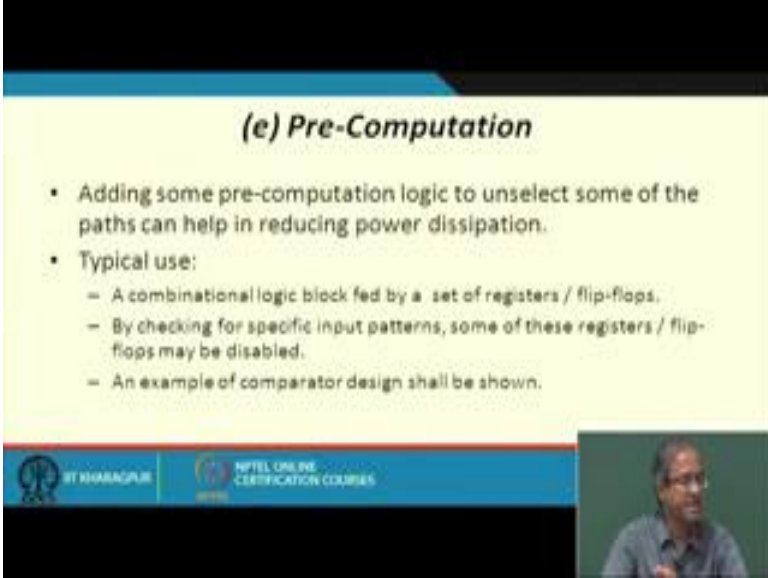


So, take a rule of the thumb. Take another circuit with another level. So, same thing will happen here also. So, if you make A B C D all change states together there will be Glitches happening, but this same function you can also implement like this. Here the delay from all the inputs to the outputs are all balanced. They all are equivalent to 2 gate delays, but in the earlier case what was happening, you just try to understand that one of the signal was coming faster, the other signal was coming slower. Because it was coming slower and this was coming faster there was an overlap. Because of that overlap this Glitch was happening, but here if you can make it balanced, then that kind of I mean unequal delay paths will not happen. So, rule of thumb is if you have a chain kind of a

structure and many gates in cascade, where some of the gates are connecting directly, this is not a good structure with respect to Glitches. There is bound to be several such Glitches here.

But if you implement the same circuit as a balanced tree structure, this will be having fewer Glitches. So, even without a formal Glitch analysis, you can analyze this circuit in terms of this topology the structure, and you can say that if we can structure it in a way where it is like a balance tree, Glitches will be less in number.

(Refer Slide Time: 06:52)



**(e) Pre-Computation**

- Adding some pre-computation logic to unselect some of the paths can help in reducing power dissipation.
- Typical use:
  - A combinational logic block fed by a set of registers / flip-flops.
  - By checking for specific input patterns, some of these registers / flip-flops may be disabled.
  - An example of comparator design shall be shown.

ST KRISHNAPUR | NPTEL ONLINE CERTIFICATION COURSES

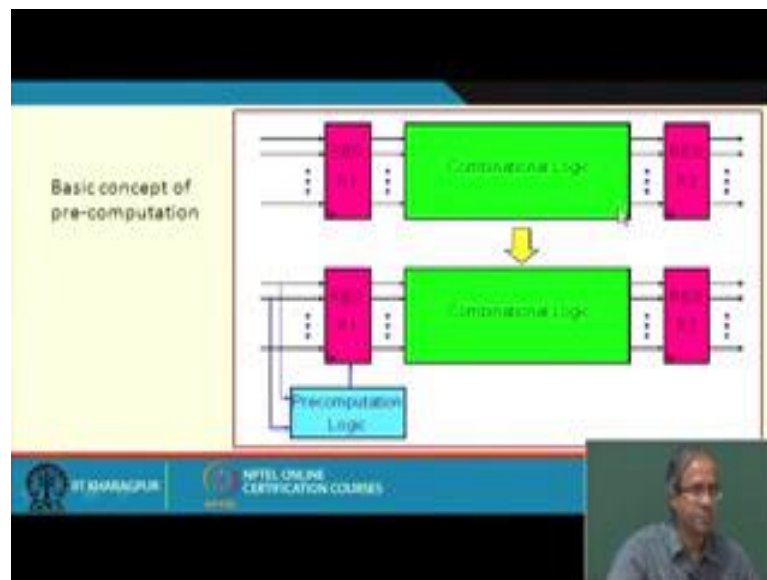
Next we look at a very interesting concept it is called pre-computation, let me just give you the motivation behind this approach, suppose we have a functional block, we will take an example later, a functional block the inputs are coming, the block is computing something and generating the output, now depending on the kind of the functional block, the total time you take to generate the output can be unequal. Like the example that I will take consider a voltage or 2 inputs coming towards the circuit is a magnitude comparator. It compares the magnitudes of the 2 numbers which are coming; you assume them to be binary numbers. And the output will be let us say 0 1 indicating less than or greater than something like that or maybe another one, whether they are equal.

Now, let us see suppose I compare the most significant bits of the 2 numbers. So, one of them is 1 other is 0. So, if one of them is 1 assuming to be unsigned, definitely that number will be greater than the other. So, do I have to look at the remaining  $n - 1$

numbers if they are n bit numbers? No just by looking at one number, one bit I can definitely say that this number is greater than the other. So, if I can take that decision early enough, then I can disable the remaining computation. This is what pre-computation is all about, some of the common cases. So, I am assuming that this is a common case. If some of the common cases can be detected early, by using some special circuits, then the output of that circuit can disable the main functional block from participating in the calculation if that condition holds which means. It is not required. So, unnecessary transitions are disabled. The idea is this. So, let us see.

So, here we act some pre-computation logic to disable or un select some of the paths which can help in reducing power dissipation. So, a typical use we shall be taking an example of a comparator. Typical use goes like this. So, I have a combinational block, whose inputs are coming from a register say. So, I can look for some specific input patterns as I said, and if some patterns are found we can disable loading of these registers. So, that new values will not be loaded. Thereby transitions in the combinational logic will be avoided. So, here we show the example of a comparator as I had said.

(Refer Slide Time: 10:16)

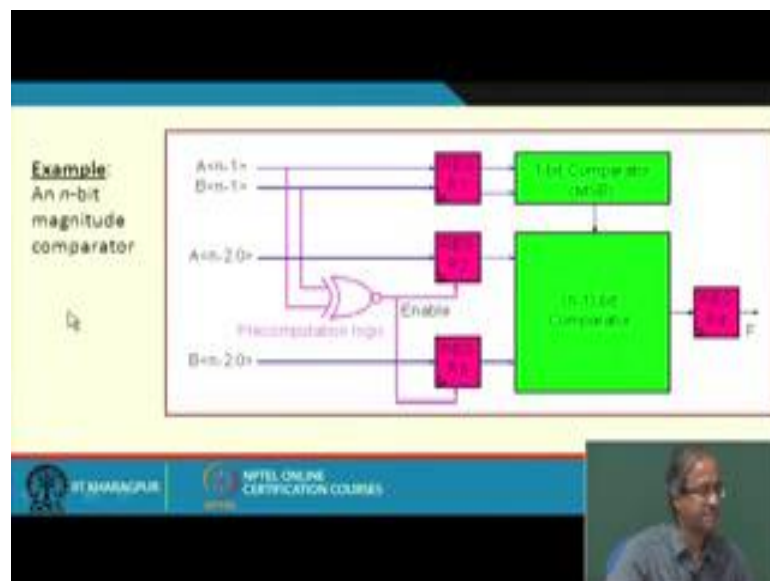


This is the general block diagram first. Let us say this was my original circuit. There is a combinational logic, there is an input register there is an output register. So, typically when things are implemented in a pipeline, we have circuits that look like this. Where

there is a clock signal that loading this register R 1. The data value will be coming to here, the results will be computed and another clock will come which will load the result in this output.

Now what we are doing here is that we are adding some extra logic here. We are looking for some specific input patterns; this is our pre-computation logic. So, if some specific patterns are there, then we can disable the loading of the resistor otherwise we will load it as usual. So, this will be like a clock gating, where normally clocks will be coming, but if some specific patterns are there, we do not need this extra computation to be carried out. Which means if I do not load this register, the outputs will not be making any transitions, which means within the combinational circuit also there will not be any transitions.

(Refer Slide Time: 11:52)

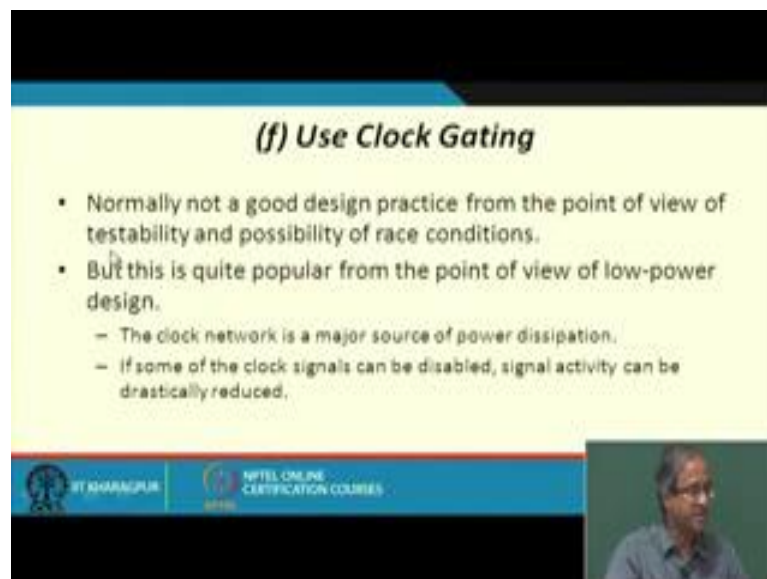


So, I am avoiding dynamic power consumption here. So, the example of the comparator goes like this. So, I have 2 numbers A and B from 0 to n minus 1, n bits. So, what I am saying is that, I take the so these numbers are all being fed into registers. So I am assuming this a n minus 1 and b n minus 1 are being fed to a separate register. The remaining n minus 1 bit of a is being fed here, remaining n minus 1 bits of b is being fed here. So, what I am saying is that, we compare the most significant bits. So, how do you compare? We have put an exclusive nor gates.

So, what does the exclusive nor mean? You see if these bits are 0 0 or 1 1. What is 0 0 or 1 1 means output of the x node gate will be 1. Only for that case we have to look at the remaining n minus 1 bit to find out which one is larger, but if we find both of them are not equal either 1 0 or 0 1 which means the output is 0. Then we can have a special one-bit comparator, just compare these 2 bits. And that result can be sent to the output directly. Without involving this n minus 1, bit comparator. Only if this bit is equal then only you activate this. So, when this bit are not equal, we are not loading any new value into these 2 resistors R 2 and R 3, thereby we are stopping any kind of transitions from happening in this part of the combinational circuit. So, we can reduce the power consumption quite significantly.

This is what I said if the 2 most significant bits are different, then there is no need to look at the other bits. So, for any kind of circuit where this kind of condition can be determined, you can use this pre condition logic or pre-computation logic to selectively disable the inputs.

(Refer Slide Time: 14:20)



**(f) Use Clock Gating**

- Normally not a good design practice from the point of view of testability and possibility of race conditions.
- But this is quite popular from the point of view of low-power design.
  - The clock network is a major source of power dissipation.
  - If some of the clock signals can be disabled, signal activity can be drastically reduced.

NPTEL ONLINE CERTIFICATION COURSES

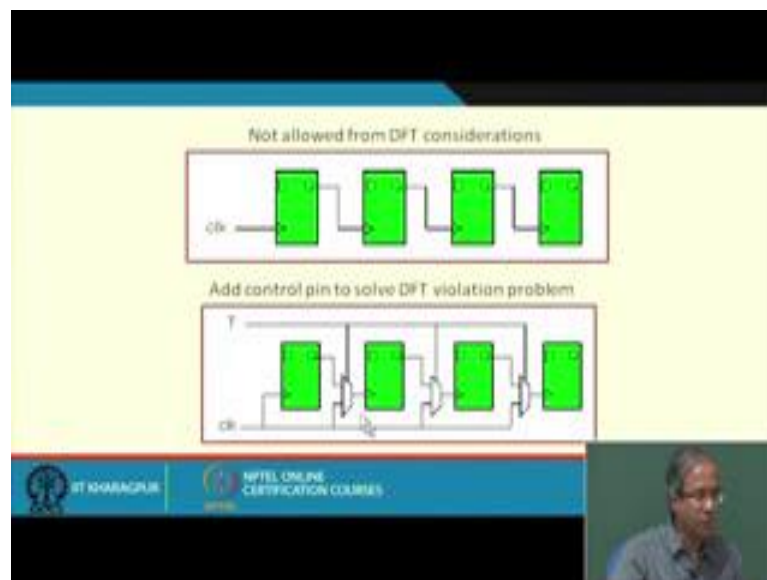
Clock gating already you know we mentioned earlier also, that we sometimes selectively stop the clock from reaching some of the flip flops or registers. So, that unnecessary transitions are avoided. So, as I had said also that this is not a good practice from the point of view of testability, and also race conditions may happen, but from low power design point of view, this is quite effective. Primarily because in a typical chip the clock



network is the one major source of power dissipation, because every node of the clock network makes 2 transitions in every clock period in every cycle.

So, that is the network where every node has the highest values of alpha activity. So, if we can disable some of these clock signals, then the signal activity can be reduced. So, targeting the clock network is very justified in the sense because those are the nodes where maximum activity is happening. So, if we can disable the part of clock network which is not correctly being used, we are effectively reducing the signal transitions quite significantly.

(Refer Slide Time: 15:49)



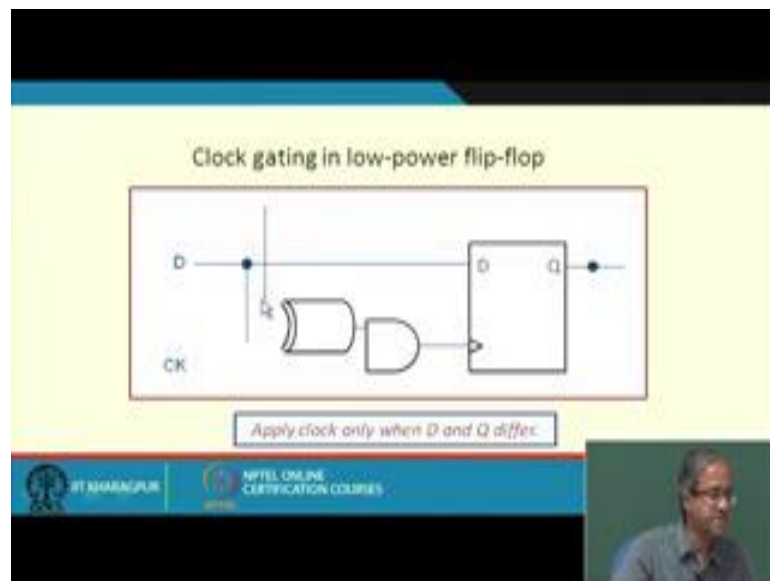
So, let us look at a simple binary counter here. This is binary ripple counter. The output of the flip flop is feeding the input of the next flip flop. Or this is like a shift register not a counter. This is the shift register the output is feeding the next one.

Now, this kind of a shift register is difficult, or this kind of a structure it is over the output of a flip flop is fed to the clock input of the next flip flop. This is not allowed from testability considerations designs for testing. Because for DFT when you want to put the flip flops in a scan chain, you need the clock signal to be fed to all of them. So, if you have a circuit structure like this, this is not very convenient from DFT point of view. So, what you do, you add some control pin and some extra hardware, to make it look something like this.



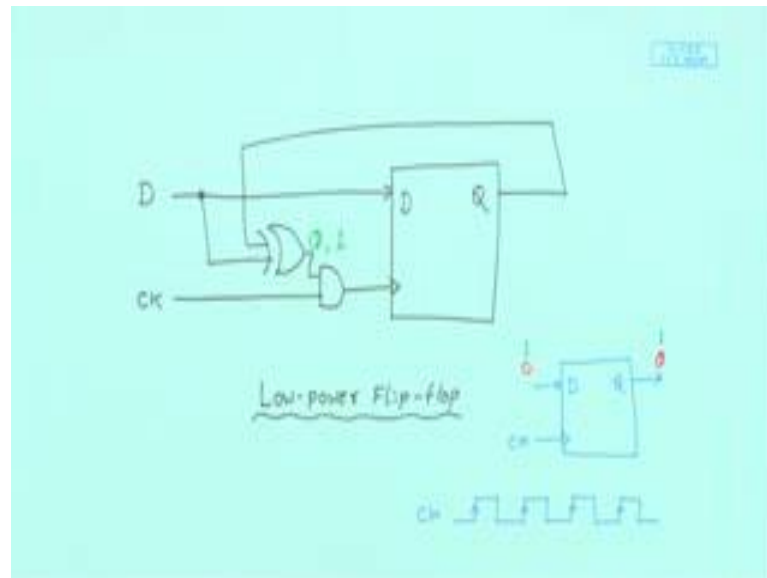
What you do? You add some additional multiplexers here, and clock you feed to all of the inputs. So, that during the testing mode you can make the clock to feed all the inputs right. And this D inputs are coming from somewhere. So, D I am not showing and this T will be the extra additional control pin. This will can be used to control the multiplexer. So, when required you can enable the clock. So, when required you can disable the clock. So, these are some very specific kind of structures if this kind of structure appears in a design then only you can use this.

(Refer Slide Time: 17:46)



This is an interesting structure. Like here 2 lines are missing anyway. Some of the lines are not showing up. See this this clock is being fed here. This D is being fed here. See here I am just showing I am just drawing this diagram again.

(Refer Slide Time: 18:19)



So, that. So, the circuit is like this I have a flip flop. So, from external input D is coming here. This is my output. So, what I do? I add some additional circuits here and XOR gate. Then an AND gate. The output of the AND gate is feeding the clock signal. Now this clock signal is feeding here and this XOR gate one of the input is fed from D the other input is fed from Q. So, why I do this? So, this is a flip flop, which I call a low power flip flop.

Let us try to justify this name why low power. See normally in a flip flop what happens, see I am just showing a side by side a normal flip flop. A normal flip flop will be having a D input; it will be having an it will be having a Q output. And it will be having a clock. Functionally these 2 are equivalent, but this structure consumes lower power. Why? Let us understand. In this circuit you see clock is a periodic signal, it is continuously coming. So, whenever there is a clock, suppose this flip flop is activated by the positive edge of the clock. So, whenever there is a clock at every positive edge.

So, whatever I have applied to the input of the D flip flop that gets stored. So in the inside this flip flop there are. So, many gates some of the gates will be having some transitions, because this clock is changing. So, there will be some um power consumption inside, but here I am making an observation. The observation is as follows. Let us say my applied D value is 1. And my current output value is 0. The current output value is also 0. Or my input is 1; the current stored value output value is also 1.

So, under this condition if even if I store it, the output value will not change. Because it was 0 it was already 0. So, 0 will get stored again. So, if it is 1 I have applied 1 again. So, it will again get stored the same value. So, here I have made just that change, which checks whether the applied input and the current output are same or not. If they are same I am disabling the clock because I need not apply the clock because the next state will also be the same. So, you see what we have done here. D and this Q we are feeding to an XOR gate. So, whenever they are 0 0 or 1 1, then only the output will be 1, sorry 0 0 1 1 the output will be 0. And the clock will be disabled, but when they are different 0 1 or 1 0 then only you need to store it. Then the output will be 1, this end gate will be enabling the clock and the clock will come.

So, here the clock will be coming to the flip flop only for those instances where the applied D value and the output Q stored value are different. Well of course, the overhead is 2 additional gates one XOR and one and gate are required. So, this is the so called low power flip flop, which you can use in a design, to reduce the signal activity inside the flip flops.

(Refer Slide Time: 22:33)

**(g) Input Gating**

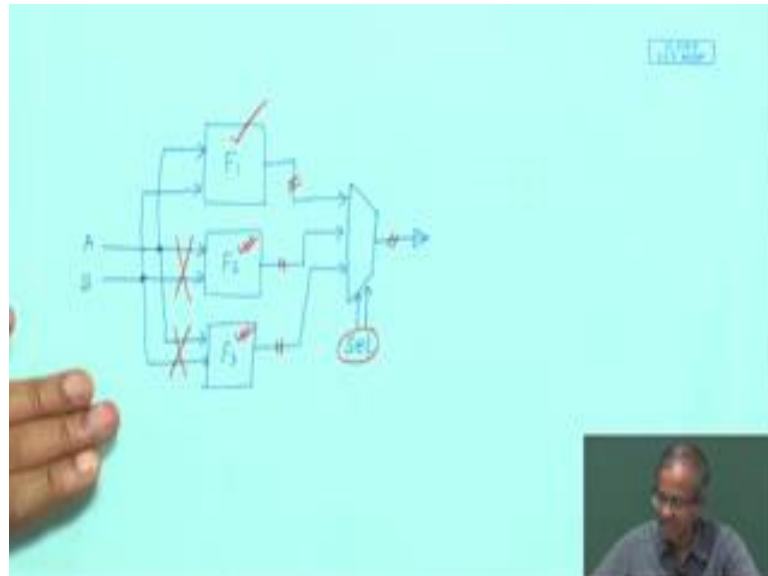
- Disable some of the inputs (i.e. prevent them from changing states) depending on specific control functions.
- Consider an example of an addition / subtraction unit.
  - Separate adder and subtractor connected in parallel.
  - Output selected from one of them using a MUX.
  - When addition is being carried out, inputs of the subtractor can be disabled; and vice versa.

IIT BHARUACHIPUR NPTEL ONLINE CERTIFICATION COURSES

Next comes the input gating technique. This is also quite simple and commonly used. Here we again disable some of the inputs; that means, we prevent them from changing. So, we shall take an example to illustrate. Like you see sometimes it happens in a data

pack in a particular circuit you have so many functional blocks. Some of the some of the operations maybe required some in some cases and in some cases we may not require.

(Refer Slide Time: 23:12)



Let us take an example. Suppose I have a few functional blocks let us say F 1. I have another block F 2 I have some other block F 3. Let us say one is adder, one is subtractor, one is multiplier. And I have 2 inputs which are coming A and B. This A is being fed to all these 3 blocks. B is also being fed to all the 3 blocks. At the output side what you do, we use a multiplexer which will take the outputs of these functional blocks. And will be selecting one of them to the output depending on some select lines. So, the select line will give the signal whether I need to do the addition or subtraction or multiplication.

So, one of them will be selected, but you see in this design there is a problem with respect to power consumption. Suppose I need to do addition; that means, I want to run block F 1. So, I apply some inputs A and B, this F 1 gets computed, result is available here. So, I apply select line this result comes out, but in this circuit what will happen because A and B are changing this F 2 and F 3 are also participating in the calculation. There is also signal changes going on inside. They will also be computing some result in the outputs, but; however, multiplexer will not be selecting them. Multiplexer will be selecting only one of them. So, the idea is, so if I know from select that I am going to select F 1, so, why do not disable the inputs of F 2 and F 3 in some way. So, that signal

transitions here and here do not occur right. This is the basic principle behind this method.

(Refer Slide Time: 25:27)

- Control signal  $F1$  selects subtraction operation.
- Control signal  $F2$  selects addition operation.
- Do not unnecessarily allow the inputs to toggle, to cause useless signal transitions.

So, here let us take an error and subtractor. Say we have an adder, we have a subtractor, the same kind of example that I took. So, in the input I am assuming that the 2 inputs that are coming A and B, they are stored in 2 registers. And the clock is used to load them. And with output I have a multiplexer, which is selected whether I want to do addition or subtraction. The output of which is again going into another register. Now here we are using some kind of gates. This can be buffers this can be tri state buffers whatever, but the idea is that whenever I want to addition, I will be enabling these buffers; I will be disabling this disabling these.



Means there will be no signal transitions occurring here which means no transitions will be occurring inside this? So, anyway I will be selecting the marks so that the output of this adder will get selected, so no problem. So, I am assuming that  $F1$  the control signal selects subtraction, and  $F2$  selects addition. So, we are not unnecessarily allowing the input of the other block to toggle. Because this will be use useless transition.

Suppose when I need to do addition, I will not be using subtraction. So, why unnecessarily make this signal toggle and make this consume additional power. This is the basic idea.

(Refer Slide Time: 27:01)

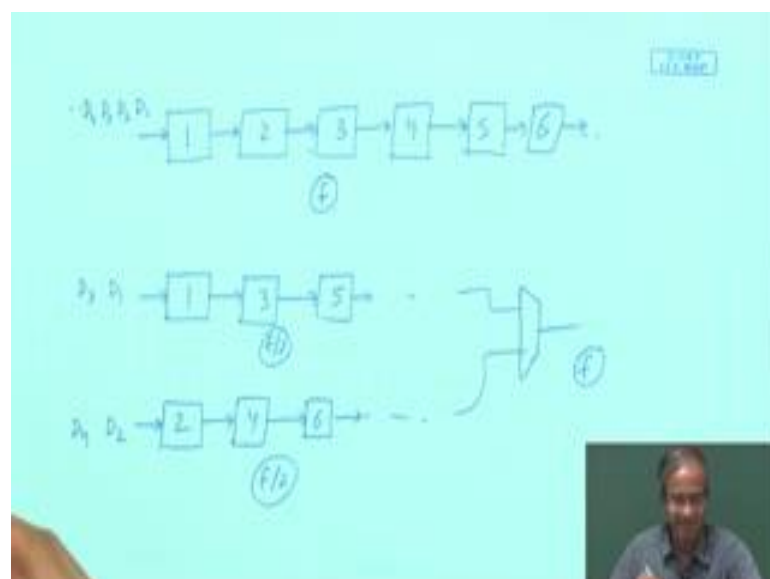
### A Miscellaneous Technique

- Design of a reduced-power shift register.
- Basic idea:
  - Power dissipation is directly proportional to the frequency.
  - We partition the flip-flops of a shift register into two subsets.
    - One enabled by the positive going edge of the clock.
    - Other enabled by the negative going edge of the clock.
    - Clock frequency is made half.
  - Apply inputs alternately to the two subsets, and also take outputs from the two subsets alternately (w.r.t. clock edges).



So, and one last technique we discuss here. This is miscellaneous technique very specific. This is a reduced power shift register. Now a shift register is normally what, it is a chain of  $d$  flip flops connected in cascade. We feed data from one side, with clock they get shifted. We take the data from the other side. Let us say I have an  $n$  stage shift register, I am clocking shift register with a clock of frequency  $f$ . Now I want to make my circuit the shift register low power. So, the principle I am following is something like.

(Refer Slide Time: 27:55)

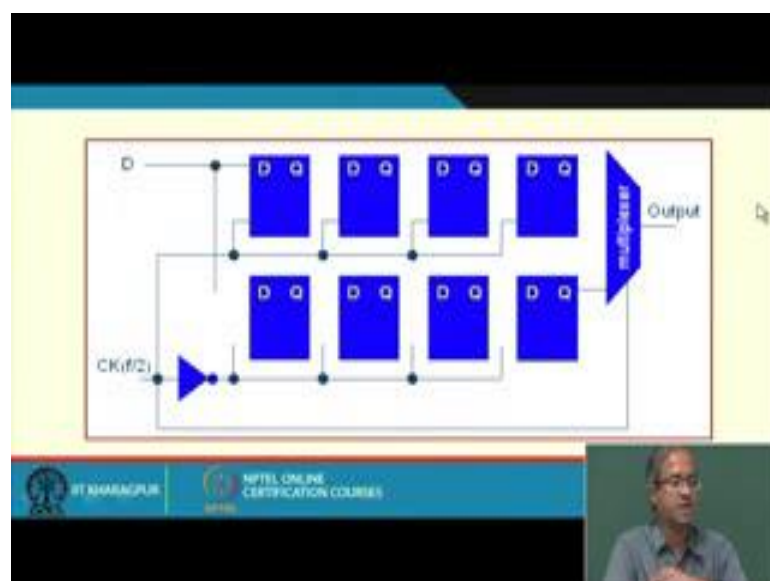


This I am splitting the shift register into 2 parts. See it is something like this. See my original shift register consisted of this stages, say stage 1 2 3 4 5 like this 6. Now I split them into 2 shift registers, with the odd number cells in one shift register, and the even number cells in the other shift register.

See earlier case I was shifting my input data, at one end and they were going through all of them one by one, but now I have 2 different shift registers. Let us say in the earlier case the data I was applying was D 1, then D 2 then D 3 then D 4 like this. Here what I do I apply D 1 here then I apply D 2 here. Then apply D 3 here then apply D 4 here alternately. And I clock this shift registers at half the clock frequency. Here let us say the frequency was  $f$ , here I am clocking at  $f/2$ , and  $f/2$ , but in the final output, if I use a multiplexer. So, I can take the outputs alternately from this and this and I can generate the output at the frequency of  $f$ . Because each of them is shifting at half this feed, but overall the data which is coming out will be double this rate, it will be  $f$ . So, the idea is like this.

I am showing the diagram. Here what I have said is that we are exploiting these principle power dissipation is directly proportional to the frequency, we partition the flip flops into 2 subsets. The first one we are clocking by the positive edge, and the other one we are clocking by the negative edge. And we are making the clock frequency half and as I said we are applying the input alternately to the 2 subsets D 1 D 2 D 3 D 4 like that.

(Refer Slide Time: 30:12)





So, the schematic will look like this. These are the 2 shift registers. These clocks are coming; this edges are missing anyway. You can just put them in. So, this clock is coming. So, I am clocking them with half the frequency  $f$  by 2. And this final multiplexer which is going out that will be taking the data from here and also here and generating output at a clock frequency of  $f$ .

So, this is one very simple technique which you can use to increase to keep the speed of the shift register at this same level, but to reduce the power consumption quite significantly. So, these kinds of adhoc techniques you can imply in various blocks of your design, some of the very popularly used methods I have discussed. And some more we will be discussing later. So, you can try and reduce the power consumption.

So, with this we come to the end of this lecture. So, we shall be continuing with some more techniques later in our next lectures.

Thank you.