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

Lecture - 62
Other Low Power Design Techniques

So, welcome back. So, we continue with our discussion on the various techniques for low power design, we have seen some methods or techniques which can be use at the level of gates or at the so called Atrial level design. We shall also be looking at some architecture then higher level techniques which we can use. So, we basically continue with our discussion what you are talking about last day and look at some other techniques that we can employ or use for reducing the power consumption in design. So, other low power designs techniques.

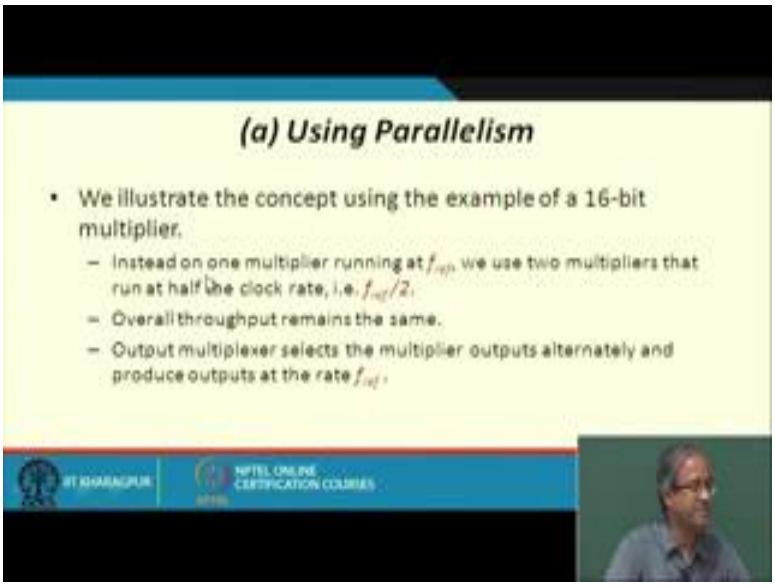
(Refer Slide Time: 00:59)

So, we look at some of the architecture level approaches here. See architecture level approach means we are not looking at the level of detail where we have the gates and flip flops, we are looking at a slightly higher level; may be at the register transfer level, we have multipliers, adders, we have a pipe line, we are thinking globally at that level we are not looking at the level of gates. So, several such architecture level approaches have been proposed while various people and some of the methods we shall be explaining the

essential idea behind it, these methods can be used again to reduce power. Some of them use parallelism, pipe lining, retiming and something called bus segmentation.

Now, you will see that many of this techniques they though require some additional hardware. So, you are increasing the cost in terms of area, or in terms of number of gates or transistors. But this is quite acceptable nowadays as I mentioned earlier also, reduction of power has become of paramount importance. Power reduction is sometimes considered to be the primary thing, only after that the other features like performance and things will come. Because you say if you suddenly go to the market and see that someone is selling a laptop and saying that it can survive 15 hours of battery once you charge it, you will be pretty impressed. So, the other features will become secondary for you may be for sometime at least, all right.

(Refer Slide Time: 02:56)



So, let us look at these methods, so what they are. So, using parallelism see these are very ad hoc approaches. So, we illustrate this concept with help of a various simple example, let us take a multiplier. See the idea is very simple; suppose in a processor I have a multiplier, the multiplier is suppose to multiply 2 number obviously, and I need or I have some requirement of the required throughput like I want some certain n number of multiplication to be carried out within the time T let say; this is what my requirement is.

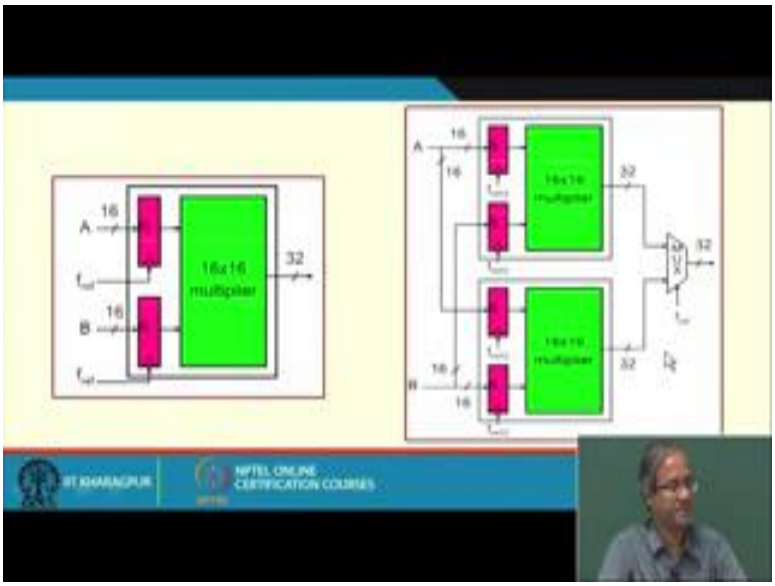
Suppose I am building a DSP processor, which is suppose to compute some operation on some signals which are coming in real time, image audios some kind of signals. So, I

know what kind of or what is the number of multiplication operations that are required to be carried out per unit time or within a time T. So, I know the delay.

So, now I can explores several design alternatives; like let us say I can say that well I have a multiplier that is fast enough, which can sustain the throughput that is expected. So, this fast multiplier is working with some frequency let us say f or f reference, now I say that will let us do something like this, let us replicate the multiplier well you are not that much worried about increasing the idea, because nowadays we can put more harder on the chip that is very easy now a days. So, let us use 2 multipliers instead of one, and let us share our load between the 2 multipliers, and what we are doing? We are reducing the frequency by half; say earlier we are carrying out every multiplication in time delta, now each of these multipliers will be carrying out multiplication in time delta by 2, but because there are 2 multipliers the total throughput will still be delta, because in this time delta you are finishing 2 multiplications. But because you are reducing the frequency by half, the impact of the power will be much more significant. So, overall power dissipation will go down this is the idea.

So, whatever we have mentioned is written here, I have mentioned instead of this half instead of one multiplier running at a frequency f ref, we use 2 multipliers so replicated and they run at half the clock frequency. So, overall throughput naturally remains the same, and there will be a multiplexor the output side that will be taking the output alternately from the 2 multipliers, and will be generating the results at frequency f ref.

(Refer Slide Time: 06:15)

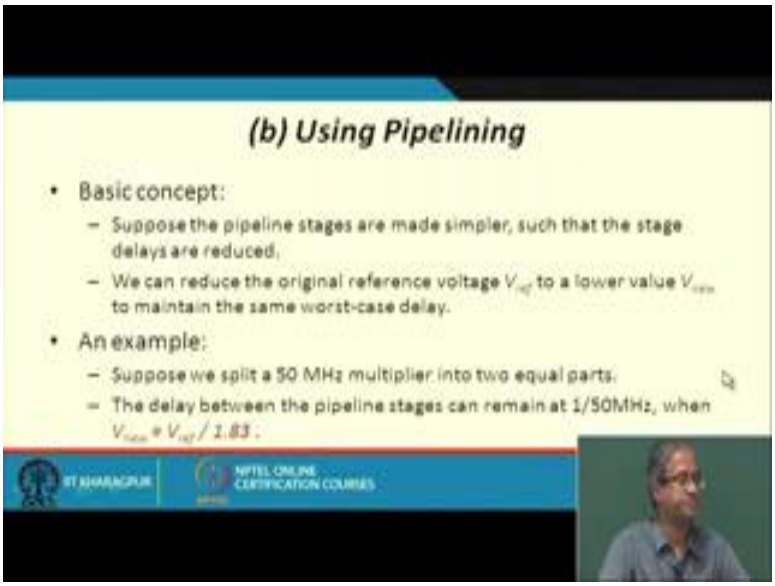


So, something like this, this was my original design, so where we had let us 6 multiplier 2 numbers A and B were coming loading to register, they were clocked with the frequency of f ref. So, results to generated 1 product in time 1 by f ref. But now what I am saying is that we use 2 multipliers with again registers separating them, and their clocked by a clocked was frequency is half.

Let say one of them we are loading by the leading edge of the clock raising edge, and the other we are activity of by falling edge of the clock. So, we are supplying 2 sets of data every clock. So, one of them were feeding to here other were feeding to here; because these 2 multipliers are no slower data coming at a f ref by 2. So, the input data transitions are also happening at a slower rate, so number of transitions here also will be slower, so products as the generated. So, again in the output side you are taking out the products one at the leading edge one at the falling edge, one from this one from this.

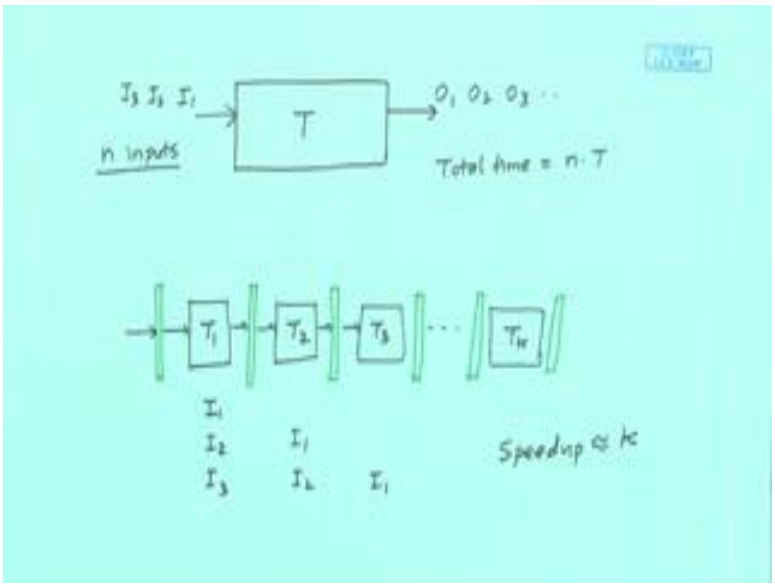
So, with respect to the final output the throughput of the circuit remains the same, but with respect to the individual multipliers there are far few are number of transitions and the power consumption on the average will be less, this is the basic idea in this approach.

(Refer Slide Time: 08:01)



Second concept involves pipe lining; let me recall or review the concept of the pipe lining for those who may have forgotten the basic concepts. So, I am just explaining the basic concept of pipe lining first before I go through this.

(Refer Slide Time: 08:23)



Suppose I have some computation that takes a time let say T. So, I have a set of input data, I apply them one by one I 1, I 2, I 3 and I get the outputs O 1,O 2, O 3. So, if there are the n number of inputs, so the total time will be n multiplied by T. Now the concept of pipe lining says that I am splitting this computation T into several parts. For the timing

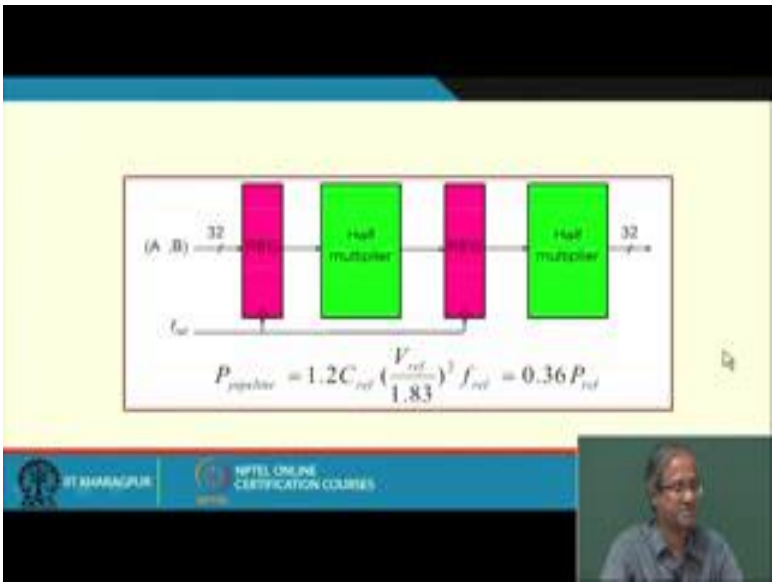
let us assume their all of equal size means they take equal times. Let say there are k number of stages I am dividing into T 1, T 2, T 3, T k; and for the pipe lining of course, will need some registered stages in between like this. So, data will be for that one end they will go like this.

Now, the basic idea is that when this input I 1 comes, first I 1 will be come into T 1 it will get executed, but after sometime T only finished. So, this I 1 in the result will go to T 2. Now T 1 can start with I 2, then next step I 1 will go to T 3, then I 2 will come here and I 3 will come here. So, in this way we can get overlapped execution. So, when n is large we can get us speed up, which is approximately equal to k equal to number of stages. So, by using pipe lining we can have speed up, we can have more computation per unit time. But here we are talking with respect to power consumption; let us imagine a scenario like this, at we had a circuit block that was good enough to meet our dead line we do not want to make it faster.

Now, suppose we break it into 2 parts in a pipe line or k parts in general, we make it in a pipe line. So, if we make it in a pipe line then it can be run faster, but we do not need it to run faster. So, now, what you do we make a pipe line all right, but you run the pipe line at a much slower frequency, such that my over all throughput remains the same as the original, but my average power consumption drastically reduces this is the basic idea.

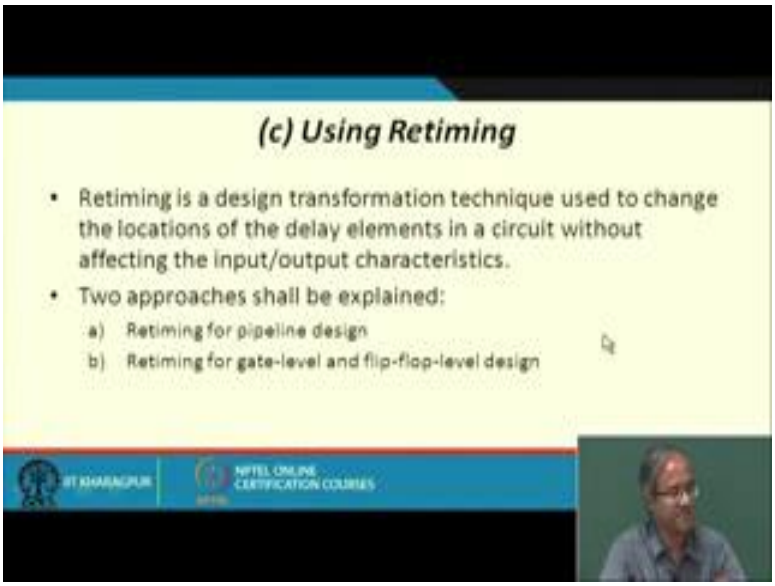
So, here you are saying that we are making the pipe line stages simpler; just like you do divide a computation into simpler sub computation, and each of this stages we can either reduce the frequency or we can reduce the voltage power supply voltage. So, reduction of power supply voltage is also very important way to reduce the power consumption; because you recall dynamic power is proportional to this square of the voltage. So, the different pipe line stages you can reduce the power supply voltage also to make it a little slower and of course, to reduce the power consumption (Refer Time: 12:39). So, this is just a very specific example you will just ignore the figures for the timing, this is for a particular case study. So, with respect to the original voltage, the voltage was reduce to by 1.83 times. So, when it is divide into 2 equal parts.

(Refer Slide Time: 13:01)



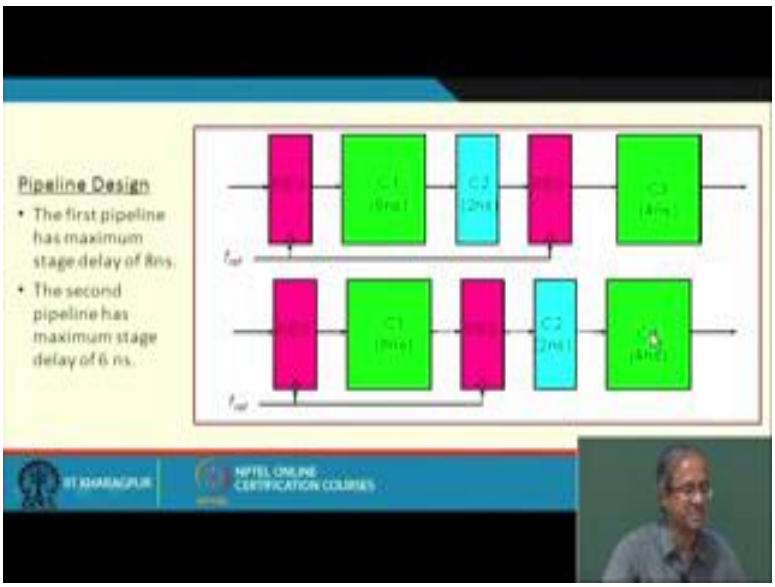
So, it was something like this. So, there was a multiplier, we divide the multiplier into 2 parts first half and the second half in a 2 stage pipe line, and what we do? We run the pipe line at the same frequency, but the voltage initially it was V ref, now it has become V ref by 1.83. So, if you compute the power consumption for this new pipe line, new design, the power consumption was about 36 percent of the original power consumption; because it is square of the voltage proportional to square of the voltage and you are reducing this voltage, this is the essential idea right. So, we are splitting a computation into smaller sub computation in a pipe line, and we are reducing the power supply voltage of these stages, their by reducing the overall power consumption.

(Refer Slide Time: 14:03)



Retiming is something which we talked about earlier, but retiming we had use to earlier we talked about to improve the performance of a design, but now here we are talking about retiming for improving the power dissipation performance. So, here we shall be looking at retiming for a pipeline design and also retiming for gate level design with flip flops just we shall see some examples.

(Refer Slide Time: 14:35)

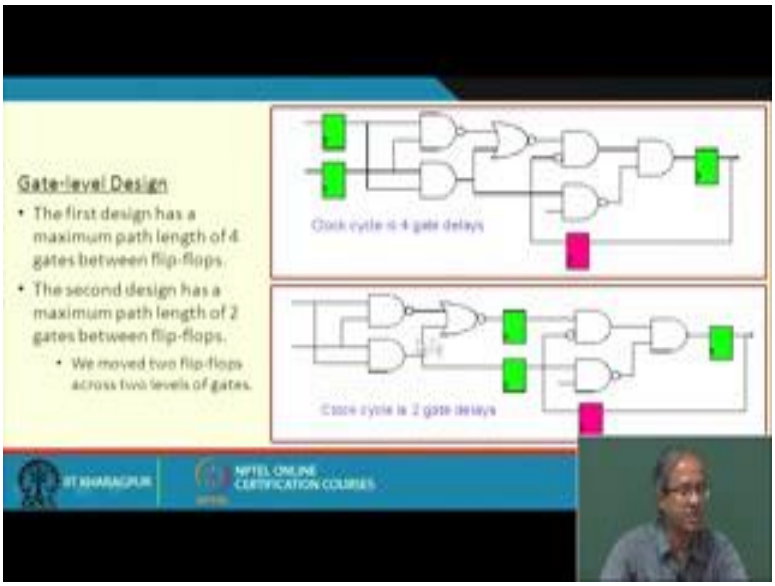


See here you look at a pipeline like this, there are 2 registers, there is some combinations circuit in between with delays of 6 nano signals 2 nano signals total is 8; and here it is 4.

So, in this pipeline the clock frequency has to be search that the time period should be at least 8 nano signals, plus of course, it register degree set up whole times all those things will be there; but if you can shift or move C 2 here like shown here, now your pipe line becomes balanced 6 nano second and 6 nano seconds. Now what I am saying is that here our objective is not to make the pipe line work faster, suppose I already had a design like this which was good enough to meet my timing requirement.

Now you are saying that we are moving around to make it even more efficient, but this 8 nano second is good enough for us, but now we have a scenario where you can work with 6 nano seconds. But now what I can do I can again reduce my power supply voltages in this blocks, so that I make these blocks a little slower, so that again I can achieve that time of 8 nano seconds, this is the basic idea that I use retiming to make my pipeline more efficient, but to restored or to retain the original timing requirement, which is good enough for me I make the blocks slower by reducing the power supply, and if I make it my power will also go down this is the idea.

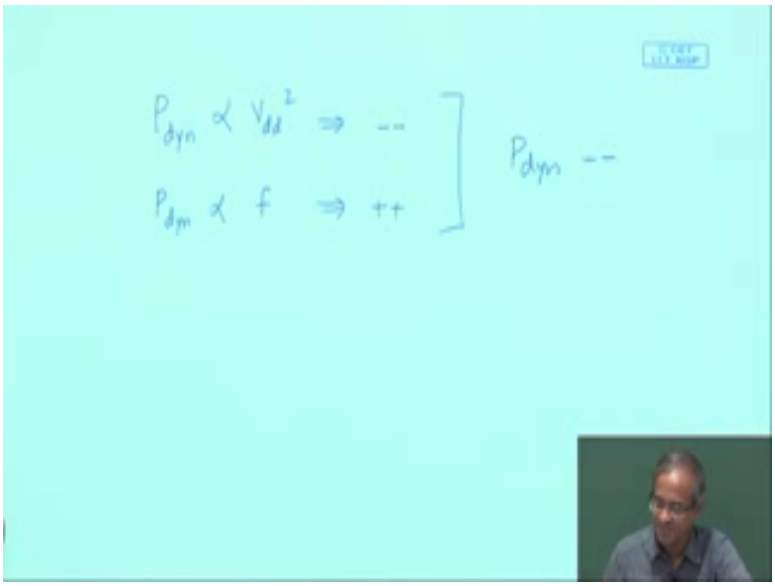
(Refer Slide Time: 16:27)



So, another example I am showing for a gate level design. So, here suppose I have a gate level design like this some gates, with some input flip flops and output flip flop. So, there are 4 gate delays in between. Suppose what I do these 2 flip flops so move across flip flops and place it here. So, now, the delay becomes uniform 2 and 2. So, initially clock cycle was 4 gate delays, but now clock cycle is 2 gate delays. So, again the same thing

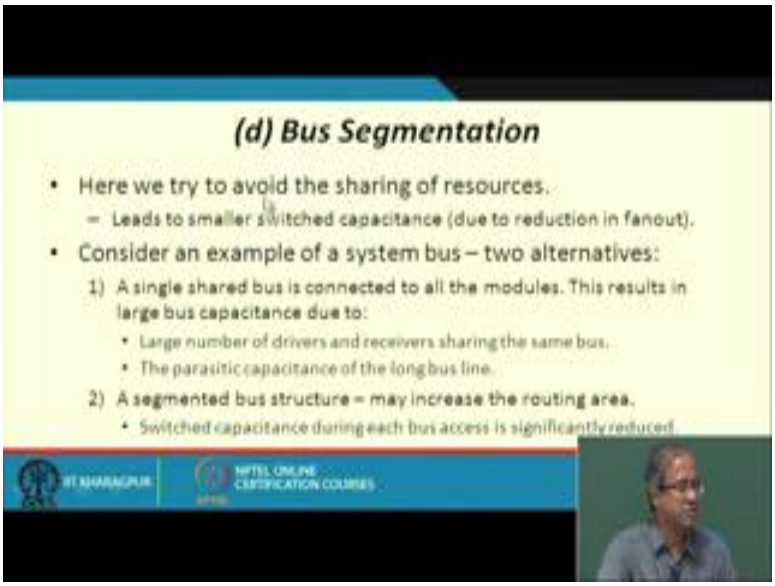
now clock can be made faster, but these circuits can be made slower by reducing the supply voltage. So, there are many miss mean approaches you can because the point you note that is that the power dissipation particularly the dynamic power dissipation, is proportional to this square of the voltage, and it is proportional to the frequency.

(Refer Slide Time: 17:23)



So, if you increase the frequency but if you decrease the voltage, the impact of this decrease will be much more because it is proportion to square of that. So, the idea is that you increase the frequency, but increasing frequency is expected to increase the power, but at the same time you decrease the voltage; decreasing voltage will be having a greater impact of the power. So, over all your dynamic power dissipation will decrease right this is the basic idea.

(Refer Slide Time: 18:14)



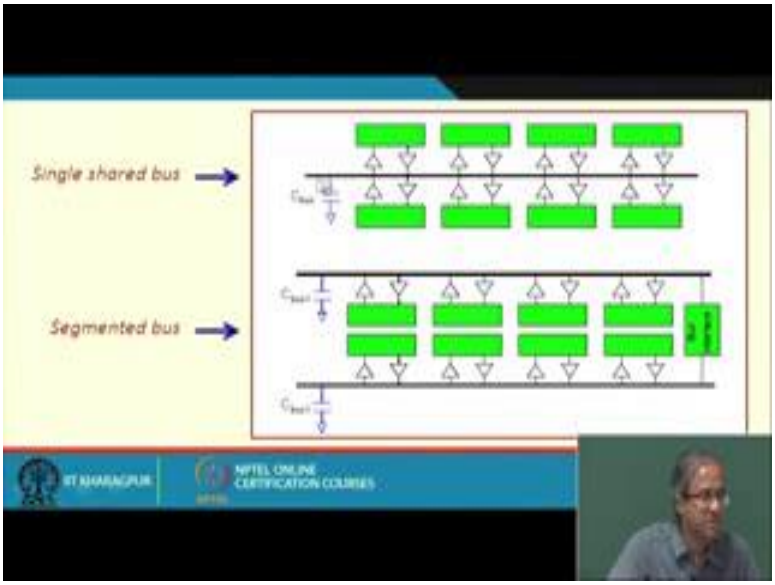
Fine and here the last method we talk about this is quite interesting, this is with respect to the bus, we are trying to segment a bus. We see what is the bus. In many designs, in particular the complex design like a processor where there many functional blocks they want to communicate among themselves. We define a bus a common shared set off lines through which multiple blocks can communicate among themselves like. Now in a CPU you may be knowing in a processor, they may be 1 or multiple such buses between the registers and the alus, alus and the memory and so on the buses are connect to several function units and they can communicate.

Now, in this approach what we are saying is that, we are looking at the buses look at a bus, and see whether we can split or partition a bus into a set up simpler buses; this is the basic idea. So, if you can do that this is called bus segmentation. So, here we are trying to avoid the sharing of resources as much as possible. So, I shall be explaining this, this will lead to smaller switched capacitance, because of reduction in fan out I shall explain this with an example.

The example that we take we explore 2 alternatives; first alternative says we have a single shared bus, a single bus which is connected to all the modules; this will result in a large bus capacitance because for each of the modules they will be a driver and receiver and because there is a single bus the length of the bus will also be quite long resulting in larger parasitic capacitance. A second alternative says you split this bus into 2 smaller

buses. So, bus capacitance will reduce and also less number of drivers and receivers will be sharing 1 bus.

(Refer Slide Time: 20:44)



So, let me explain this with the help of an example take a scenario like this, the first case you say there is a bus which is indicated by this solid line, there are 8 modules which as sharing the bus, there are drivers and receivers sending and receiving drivers and buffers with all the modules. Let say when this particular module once to sends some data to this module, then this driver will be activated and this receiver will be enabled. So, this bus is being shared by all the modules. Now naturally the length of the bus will be long enough because it has to connect all this 8 modules. So, all though I have shown it like this in an actual layout, they can be placed to one after the other in a long stretch, the blocks may not be placed one below the other in a real layout.

And the other thing is that if you look at a particular block let say this driver, the output of this driver is driving so many buffers 1 2 3 4 5 6 7 and also itself 8. So, fan out is 8 right and because the buses long. So, the parasitic capacitances with something like C bus. So, in the alternate design what we do? We are saying is that this bus we are splitting into 2 buses; the first buses connecting only this first 4, the second bus is connecting these 4 of course, there is an interface to communicating between the buses whenever required. So obviously, will be doing this bus partitioning by looking at the data transfer requirements; so most of the data transfer that is data taking place you want

to put them into one bus, between buses number of data transfers will be less that is the idea.

Now, if we do this, first thing is that the size a length of the each bus will be less because it will be connecting less number of modulus. So, the individual parasitic capacitances will be much less smaller than C bus. Secondly, now each driver earlier it was driving 8 loads a fanout of 8, now it is having a fanout of 4 plus this bus interface much less. So, in the parasitic capacitance will be less C will be less. So, as you can see by doing this kind of simple partitioning were able to reduce not only the parasitic capacitance, but also the load capacitance, thereby we are able to reduce the dynamics switching power to a great extent. This is one of the very commonly used and popular techniques for reducing power.

Now, we have looked at a number of architecture level techniques for reducing power. In our next lecture we shall looking at some algorithmic technique, algorithmic technique means we have not yet designed our circuit, we do not have an architecture as yet, we are yet to design. So, even at an algorithmic level or a behavior level what are the techniques that you can follow that will ultimately result in a reduction in power, those are a few thinks that we shall be looking at an explaining in our next lecture. So, for now let us stop here.

Thank you.