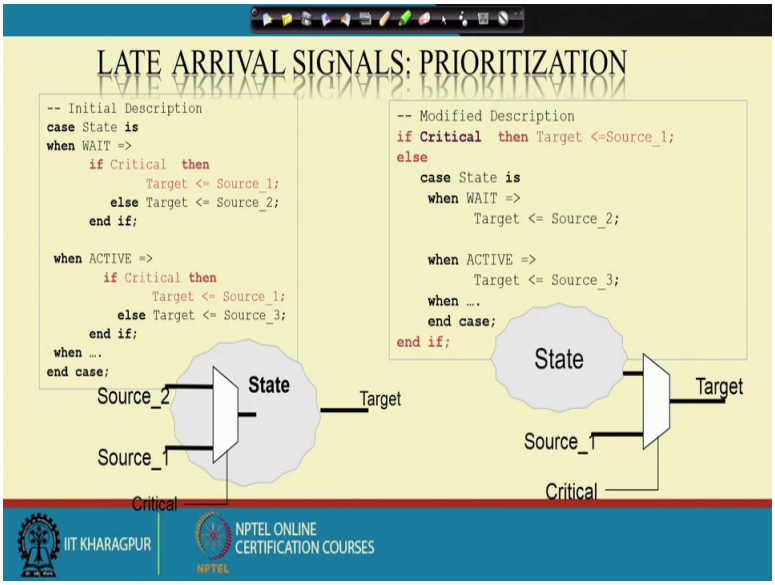
Architectural Design of Digital Integrated Circuits
Prof. Indranil Hatai
School of VLSI Technology
Indian Institute of Engineering Science and Technology, Shibpur, Howrah

Lecture - 07
Tutorial - I

Welcome back to the course on Architectural Design of ICS. So in the last class we have seen this; that means, in one particular circuit how we can implement this 1's complement as well as t2's complement.

(Refer Slide Time: 00:30)



So, now we will see; that means, some of the architectural; that means, changes to; that means, to handle the problems how we can change the or how; that means, what are the tips or what are the tricks we can do so that we can avoid some of the; that means, critical condition whenever we are designing the circuit ok. So, this is something; that means, apart from the architecture to; that means, algorithm to architecture mapping, but this is very much necessary whenever you are working to meet the specification ok. So, suppose you here consider one case ok, where this is one algorithmic description which is given to you.

So, that means, this algorithm is says that case state is when wait state, then the condition is that if this critical then target will be selected from source 1 otherwise if it is not critical then target will be selected as source 2. So; that means, in one particular; that

means, there are 2 state when there is wait state and active state. In wait state, if the condition is critical at the time the source 1 will be selected to the target otherwise source 2 will be go; that means, selected as the target.

If when that is inactive state at the time this critical will be set to source 1, otherwise target will be set to source 3 ok.

So that means, here if you see that in any of the case in any of the state whenever I find that critical at that time this target is basically selecting to source 1 ok. Otherwise it is basically selecting this source 2 or source 3. In active state or in wait state, so now, this will be the corresponding circuit implementation for this particular logic ok.

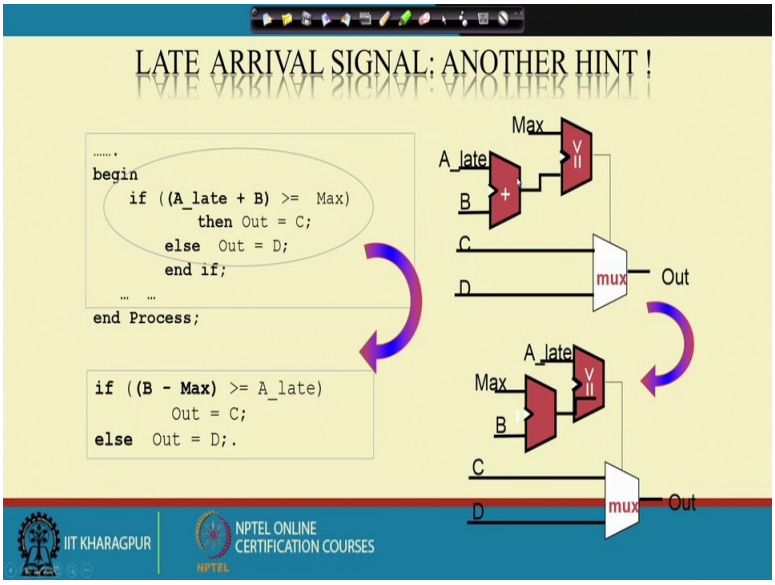
So that means, in critical condition either it has to select the source 1 or select source 2 for this wait state. Or in critical case, in active state what will be the case this source 1 and this will be source 3 for the condition of the critical. So, as I can see that this from this particular expression critical is the case where both the state it is common and it is selecting to source 1.

So, I can write or I can modify the same; that means, algorithmic description in this time manner that is if critical then target equals to target you set to source 1. Otherwise you just wait for the state to be check. That is, in wait state chose target source 2 or in active state you chose target equals to source 3. That means, at that time the critical we will select between the state or source 1. Then what is that means, what is the problem here in a critical case I need immediate action to be done ok.

So, here in this algorithm description what is the problem it has to check the in which state it is fast and then it will check whether the condition is critical or not. So that means, in critical case I have to wait for some time to check whether that is the state where that state is in wait state or inactive state. But instead of that I can do the modification in such a way, so that whenever I am getting critical case automatically this source 1 will be set to as in both the thing in both the state the target is you selected to source 1 only then that that is why I can only select this critical equals to target which is mentioned as source 1.

In other case you just chose; that means, whenever this condition is not critical at that time you wait and find out in which state it is and then you select the corresponding target as source 2 or source 3 ok. So this method is known as prioritization ok.

(Refer Slide Time: 05:57)



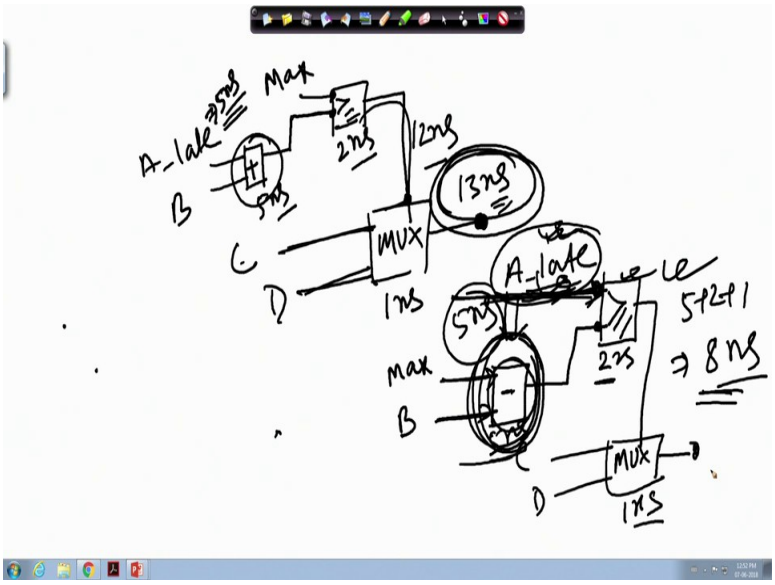
So, then another example of this later I will signal; that means, suppose one signal is coming or appearing to the corresponding critical circuit lately, so how we can handle those situation ok. So, then consider another example of that. So in another example of that here if you see this another algorithmic; that means, description it is given to you begin, if a minus A _ate plus B greater than equals to max, then out equals to C else out equals to D.

That means, A_late means a is the signal which is coming lately or which is arriving to this particular logic lately it has to be added with B the value will be then check with max if the max value is; that means, if this condition satisfied then out will be selected as C, otherwise out will be selected as D. So, what will be the circuit implementation? If I just implement this particular logic using the circuit or using the architecture if I just draw for this particular logic? So, at the time what I need A_late plus B fast. So, A_late plus B fast, then that is that has to be check with max, so that has to be checked with max.

So, those is basically 1 single input; that means, 1 bit input sorry 1 bit output depending on this particular condition if it is true; so if it is true then it will select C otherwise if it is

false then it will select D. So, there is a max which is basically the select signal is this one. So, then what is the problem in this particular circuit? Suppose for this particular case.

(Refer Slide Time: 08:04)



If I just draw the circuit again. That means, A _late and this is B then that has to be checked with max. And there is a mux depending on C and D this will be selected.

Now, consider 1 case where A_late signal is basically coming lately. So that suppose this is coming after 5 nanoseconds of delay ok. So, this is the addition operation. So, as this signal itself it is coming 5 nanosecond delay and let us consider this as 5 nanosecond delay. So that means, after 10 nanosecond I will get the signals over here and then it will be compared with max in this particular case let us consider this comparison takes 2 nanosecond. So, after 12 nanosecond this select signal is available to this particular mux. So, based on that now suppose this has 1 nanosecond of delay, so after 13 nanosecond I will get the corresponding output at this particular time ok.

So, instead of doing that can I save or can I reduce this particular 13 nanosecond which is requiring here, can I reduce it or your if I can reduce it then how can I reduce it. So, if you just go back, so if I just change the logical expression in other way. That means, this equation now if I just, that means, change it B minus sorry, this B minus max here what is what is that A _late plus B that has to be checked with max.

Now, what I am doing max either put on the left hand side and A _late I just put on the right hand side ok. So, now, what will the equation will be something like this B minus max that is greater than equals to A _late then out equals to C otherwise out equals to D. That means, the logic remains same, here if you see the logic remains same, but I just the position of A _late and max I have just changed inter changed.

So, now what is happening for that what I need B minus max, first then that has to be checked with this comparison with this A _late and there that particular select signal we will chose C or D. Now if I just again go back to that means, slide here. So, now what is happening. So here if I have just changed the logic now according to the newer logic what will happen. So, I have to consider max and B max minus B.

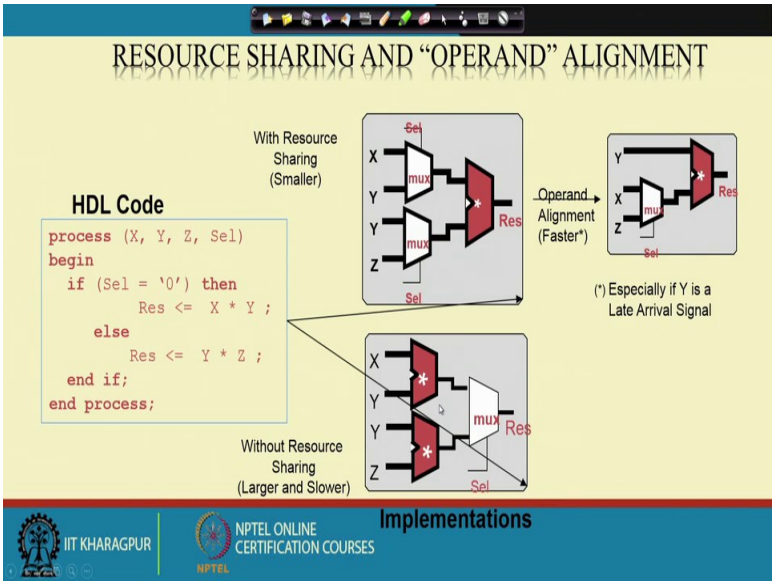
And then that has to be checked with A _late that will become to this mux this is C and D. So, this is basically appearing after sometime ok, so this has 2 nanosecond. So, whenever these signal is basically arriving at this 5 nanosecond, by that time I have already completed this 5 nanosecond work. That means, whenever this A_late signal is arriving at this particular point this 5 nanosecond this. That means, of this job is also done at that particular time. So, here also I am getting at that particular. That means, at that time this is done at within this 5 nanosecond, now this is 2 nanosecond and for this mux what I need, I need another 1 nanosecond.

So, here total to the. That means, input to the output how much time I require 5 plus 2 plus 1 that is equals to 8 nanosecond. So, initially I what I need I need 13 nanosecond, but here I need 8 nanosecond. That means, the only the change what I did is that keeping the. That means, the logic keeping the logic same only the expression I change from. That means, I interchange and what happened instead of adder here I put. That means, subtractor here adder sub tractor have both the same delay rest of the things remain same ok.

So that means, now I can achieve or I can improve the corresponding frequency of my circuit ok. So that means this is one technique whenever I am doing this algorithm to architecture mapping the algorithm was defined in such a way. So, that it is taking direct implementation of it is taking too much of time, but manually if I change something like this. So, at the time I can reduce the time. That means, reduce the things. So, here what I did I did the prioritization.

That means, because of this signal is coming lately. So, that is why I just before arrival of this particular signal the job initially I have done I am just completing the job before hand of this particular signal arrived in this particular circuit or the in this particular operation and then I am doing the rest of the operation ok. So, this is one simple example by which I can easily get the benefit of improvement in the speed ok.

(Refer Slide Time: 15:01)

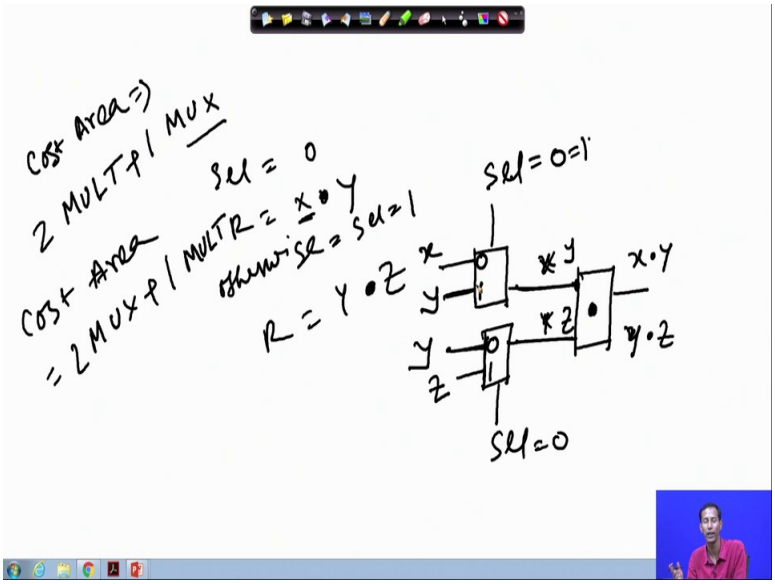


Then again if I consider another example, so this that means, this is the previous case was for how you can improve or how we can save the time ok. So, here we will see another example where I can save the area by using this resource sharing operation ok. So, this resource sharing operation suppose one this one algorithm I am having something like this process X Y X select, if select signal is 0 at that time the output will be X star Y. That means, X multiplied with Y else Y will be multiplied with Z that will be on to the output.

So, the direct that means implementation of this particular. That means, code is something like this X star Y Y star z. That means, X multiplied with Y and Y multiplied with Z is done here, and then I need 1 mux whose select signal is this sel. So, if it is 0 then it will select X multiplied with Y at the output otherwise it will select Y multiplied with Z and the output. So, instead of doing that what I can do what I can. That means, from this particular code.

If you follow in this multiplication X star Y and Y star Z both this Y is common. So, at that time can I do something ok. So that I can reduce the area more how can I reduce, the first approach is that because multiplier needs more area. So, in compare with multiplexers, so what I am doing instead of putting the select signal at the; that means, operator level if this corresponding inputs whether that is X Y or Y Z if I put the select signal. That means, if I put the mux here; that means, if I put the resource sharing on resource sharing on means in 1 multiplier how can I. That means, implement both the function. So, how can I implement, so here I am requiring only 1 multiplier then what I am doing, so what I am doing basically.

(Refer Slide Time: 18:10)



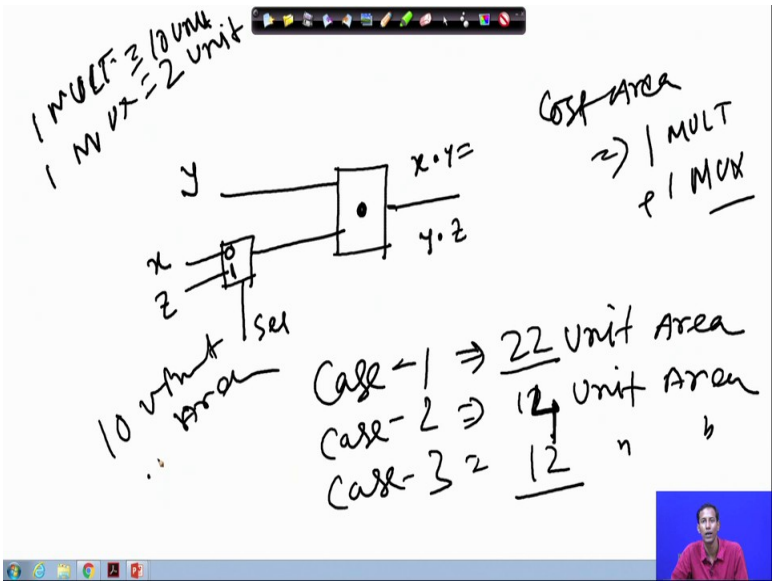
So, I have the logic select equals to 0 at that time R equals to X X star Y otherwise, otherwise R equals to Y star Z. That means, otherwise means if select equals to 1. So the direct implementation we have seen. So, the indirectly implementation if I just want to use only 1 multiplier because, there I need 2 multiplier if I want to use only 1 multiplier. So, at that time what I can do I am having only 1 multiplier right. So, and it has to be two input. So, at the very first what I need I need 1 mux and here also I need 1 mux correct. So, this is select and this is also select.

So, at the very beginning what I what is there that it needs X will be multiplied with Y. That means, here if this is X and if this is Y then X is multiplied with Y and if select equals to 1. That means, at that time what I need select equal to at that when select

equals to need when at that time what I need Y multiplied with Z ok. So that means, here select value equals to when select value equals to 0 at that time I need this is for 0 and this is for 0 and for select equals to 1 this is for 1 and this is for 1. So, whenever this select value equals to 0 at the time this will select X this will select Y. So, I will get X star Y when this value is equals to 1. So, at that time this will cross this will select Y and this will be selected Z.

So, at that time I will get Y star Z. So that means, now initially in the initial implementation what was the; that means, cost of area, I need 2 multiplier plus 1 mux right. In this particular implementation what is the cost of area that is 2 multiplexer plus 1 multiplier. So, can I reduce it more, yes I can how if you see Y is basically common in both the case ok. So, that means, now I will select whether I need X will be multiplied with Y or Z will be multiplied with Y.

(Refer Slide Time: 21:45)



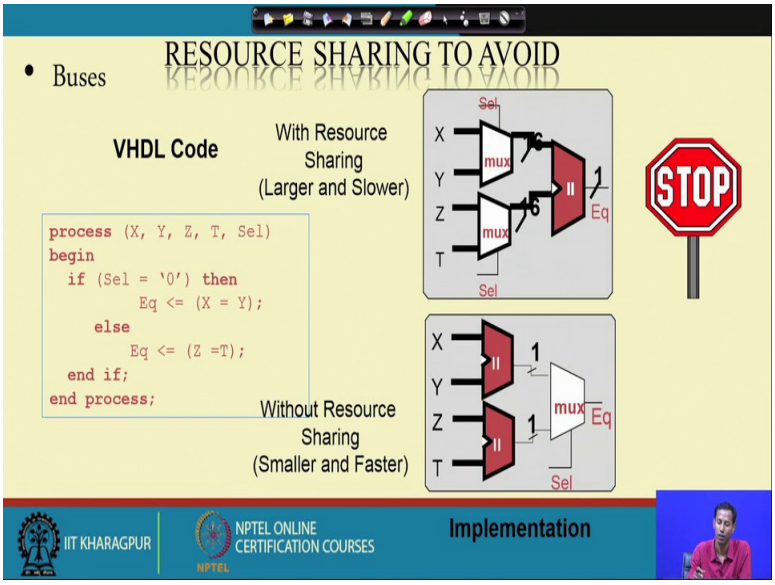
So if I just take this as this is the multiplier. So I need 1 Y is basically fixed here so here I mean I am having X and Z. So, select this is when select is 0, so at that time it will select Y if select equals to 1. So, Y will be multiplied with Z. So that means, X star Y when select equals to 0 when select equals to 1 at that time Y star Z. So, here what is the cost? 1 multiplier plus 1 multiplexer.

So that means, by this method now I can reduce the area. So this is a simple trick the code remains same or the functionality remains same, but using the tricks or using this

tips or the techniques I can reduce the cost of the area in this passion ok. So that means, if 1 multiplier cost is 1 multiplayer cost is let us say 10 unit. So, for the case 1 and for 1 multiplexer if the it is consuming 2 unit of area. So, for case 1 I need 22 unit area for case 2, I need 12 unit area sorry, not 12 that was 14 2 multiplexer and 1 multiplier 14 unit area and for case 3, I need 12 unit of area.

So that means, now from 22 to 12 so 10 unit of area I am just saving by this method resource sharing and operate alignment technique ok. So, this is just one simple things which we follow whenever we design the digital circuits ok.

(Refer Slide Time: 24:30)

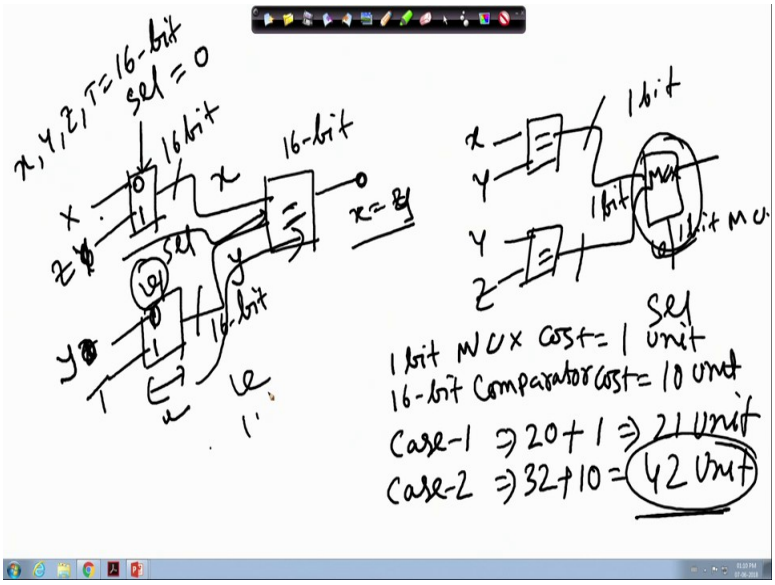


So, then here you see that with resource sharing what is happening this. That means, what is happening ok. This is another example, here you see there. That means, X Y XYZ. XYZ and this is select input and whenever this here instead of in the previous case it was multiplied here we have to check the equality ok. So, equality says that if X equals to Y then equal will be goes to 1, otherwise it will check Z equals to T and those values will be gone to the output ok.

So, the direct implementation says that X is basically first checking with Y that is scalar unit. That means, 1 and Z is basically, that means compared with T. And then that has gone to the mux if select equals to 0 then it will select this particular line if select equals to 1 then this will select this particular line ok. So that means, here what I need, I need if I consider; that means, this bus.

So, at that time what is happening I am requiring one 16 bit if this XYZ and T all these are of 16 bit. So, at that time I need the compare of comparison of this and then that will be produced this 1 and then this is gone to the mux. So that means, here without. That means, this is without resource sharing correct. Now if I just want to share the resource. That means, if I am having only if I use only 1 comparator block. So, at that time what will happen I have to choose at what time the operation will be different right? So, here I am having only 1 of this particular comparator block ok, so the same thing if I just draw the circuit so at that time what is happening.

(Refer Slide Time: 27:32)



I am now I am having only 1 comparison block right. So now, I have to select which input I have to compare ok. So, at the very beginning what I need, it has to be multiplied it has to be compared with X and Y and in the next it has to compare with Z and T right. So that means, what I need is that what I need is that I need to compare with I have to select this and then I have to compare and this. So, sorry not this one this X and Y here this will be Z this will be T, so this is 0 this is 1 sorry this is 0 this is 1 this is 1 ok. So, whenever this select equals to 0 at the time from here it will select X from here it will be select Y. So, it will be selected and it will compare X equals to whether there is equals to Y.

So, here if I consider this X Y Z and T each of these of 16 bit so at the time this is of 16 bit and this is of 16 bit ok. And this is 16 bit 2. So that means, now initially what was

there initially I was having 2 comparator, which is basically comparing here X and Y here Y and Z this is this was the direct implementation and this one was a mux. So, and this is a select line so each of this was 1 bit this is also 1 bit because this comparison result so a comparator results.

So, this will be one only and then this mux equals to this is 1 bit mux I need ok. So, if I consider. That means, 1 bit mux cost let us consider 1 unit and 16 bit comparator cost is let us say 10 unit. So, further for this particular case for case 1, where I have not used resource sharing operation so at that time what is the. That means, corresponding area here I need two of this. So, 20 plus here what I require that is 1. So that means, total 21 unit, in this particular case with when I have put the resource sharing option on.

So, at the time for case 2 what will happen here I need what 16 into 16 16 plus 16 that is 32 plus 16 here that is 10. So, total 42 unit of area. So that means, we use resource sharing of operation or. That means, option to reduce the area, but here what is happening here I am getting. That means, and in increasing the area not only the area here also I am increasing the delay, how the delay because for 16 bit multiplexer the delay will be more right.

Here what is what is there that to this. That means, any of this that will come to the in the critical path right. So, as this is 1 bit operation, so the delay for this and for to delay of this 16 bit operation that will be more. So that means, as I am using more used by putting the resource sharing option on as I am consuming more area.

So that means, I am using, that means, losing the area as well as I am consuming more power. So, it is not at all true that all the time if I put this resource sharing option on it will give you or it will. That means, gives me the benefit of area saving that is not at all true. It is the designer perspective that or this designer intuition that whether where he will use this resource sharing option on whether that is beneficial for them or not that here has to be checked.

And what logic basically there implementing based on that we have to put this resource sharing option on or off. So, these are the tricks or tips which we use whenever we design any digital circuits ok.

So, next classes in the following classes we will again see several kind of architectures or this kind of tricks and tips which are very much beneficial for digital design circuit, ok.

Thank you.