VLSI Design Flow: RTL to GDS

Dr. Sneh Saurabh

Department of Electronics and Communication Engineering
IIIT-Delhi

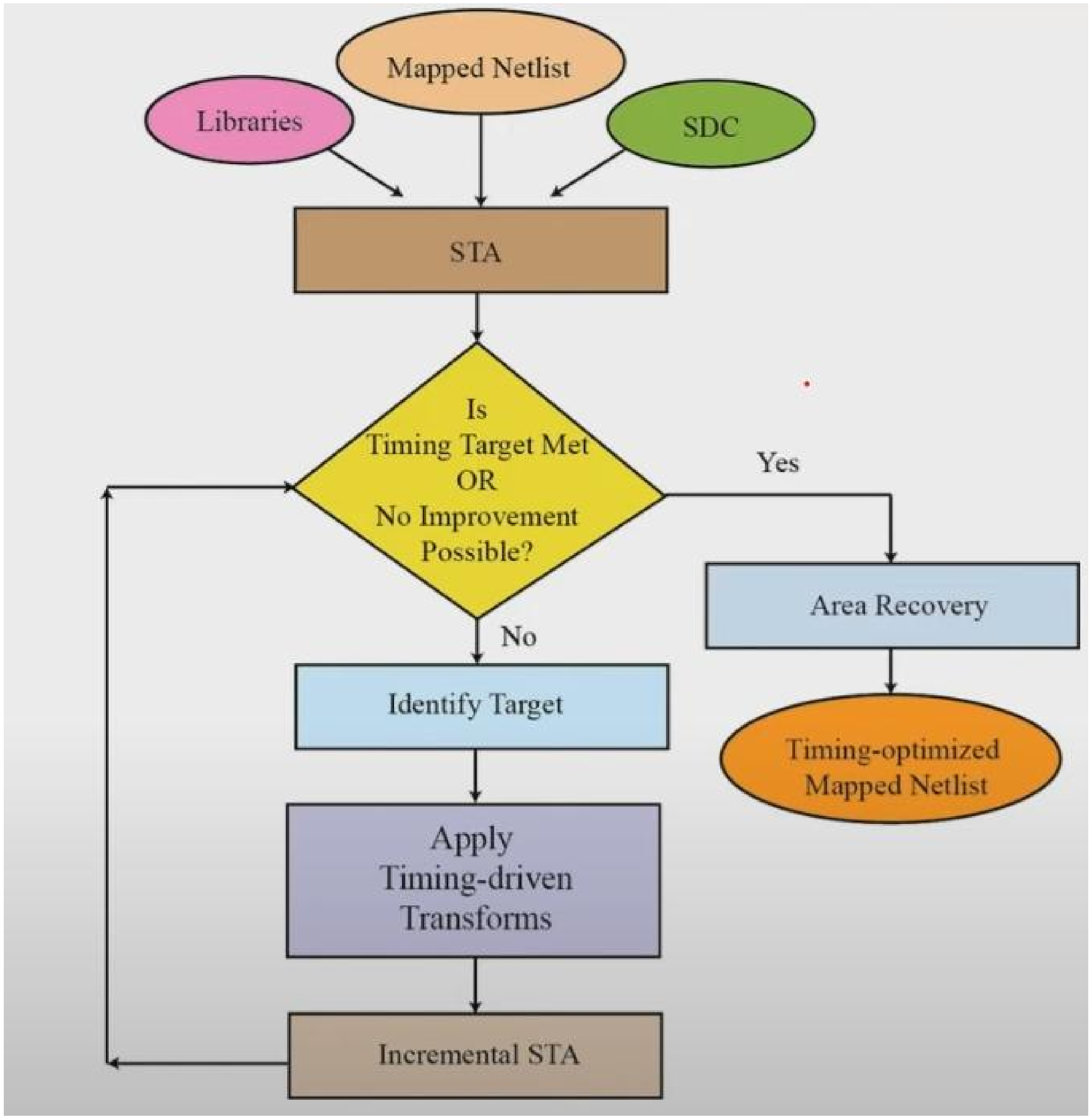
Lecture 35
Timing-driven Optimization

Hello everybody, welcome to the course VLSI Design Flow RTL-2 GDS. This is the 28th lecture. In this lecture we will be discussing timing driven optimizations. In the earlier lectures we had seen that logic synthesis consists of various tasks which are shown in this slide and in earlier lectures we had seen or we have discussed RTL synthesis, logic optimization and in the last lecture we had looked into technology mapping. So, at the end of technology mapping we get a netlist which is in terms of standard cells of the technology libraries. And in the last lecture we discussed that once we have a netlist in which the cells are from the technology libraries for which the transistor level implementation is defined, the PPA of the design can be estimated.

For example, the area of the circuit, the timing of the circuit and the power dissipation, these things can now be after technology mapping we can estimate these quantities well. And therefore, after technology mapping what we do is that we carry out further optimizations. Now since we can carry out timing analysis also after technology mapping because we have library cells and from the library cells we can compute the delay of the cells and get a fair idea of what the critical paths are in our circuit then we can target or we can optimize our design for timing. So that is what the purpose of timing driven optimization is.

So after technology mapping we know the timing of our design fairly well though the interconnects are not there in the design yet cell delay can be computed fairly well and based on that we can find what are the critical paths in our design and whether we can improve or we want to improve the timing of some of the paths and then we do some targeted timing optimization on those paths. So that is what timing driven optimization is and we will be discussing this in this lecture.

So first let us look into the flow of timing driven optimization. So in timing driven optimization what we have our design is the design is a map netlist so map netlist to netlist is in terms of cells of the technology libraries and that is why we are saying that it is a map netlist and then we also give the information to the to we give the information of the technology libraries from which the cells were picked in the map netlist to the tool and then we also give the constraints using the SDC file and then using these three inputs we can run an ST or static timing analysis . So to do timing driven optimization the first thing we need to do is to carry out timing analysis or static timing analysis and for that we need three inputs: the libraries, the map netlist and the SDC file.


Mapped Netlist
Libraries
SDC
STA
Is Timing Target Met OR No Improvement Possible?
Yes
No
Area Recovery
Identify Target
Timing-optimized Mapped Netlist
Apply Timing-driven Transforms
Incremental STA

  Now once we do STA then we can get the information of what are the critical paths in our design, whether there are some negative slacks that need to be fixed and those kinds of information we can get from ST. Now once we have done the STA then we can first check whether the timing target is met. Maybe our slack target was 0 picosecond or it was a plus 10 picosecond. Note that after technology mapping we may want to do timing driven optimization and our slack target may be more than 0 picosecond because at this point of time if we have not done physical design the interconnect delays are not cannot be computed directly and therefore if some sort of margin must be left for the interconnect delays also and that is why we may want to have our slack target more positive than 0 . So depending on whatever the slack target was or the timing target was for our model or for our design we will first check that if it is met if the target if the timing target of our design is met then we need not do anything we need not worry our design is we need not perform any timing driven optimization so it goes into this path. If the timing target is not met then what we do is that we identify targets and what do we mean by identifying targets we want to know which paths in our design are critical where we want to actually apply the timing driven optimizations .

So those those targets must first be identified the paths the logic gates or the cone of logic gates where we need to perform some optimization so we need to identify those target so once the target is identified then what is done is that the timing driven transformations are applied so some logic some optimization techniques are applied or transformations are applied to our circuit what are these transformations we will see in the subsequent slides . And once we have applied the transformation then we carry out incremental STA or incremental static timing analysis and what incremental static timing analysis does it carries out timing analysis in only a small part of our circuit. So our circuit may consist of say millions of gates but we have made some small change in say one of the logic cones in our circuit we will not want that we carry out STA for the complete design which will be taking lot of run time rather we will want that the that timing analysis is done only in the area or in the region where the timing optimization was done and where the timing timing is changing that or delay and the arrival time and the slacks in our design is changing in the portion of the design we want to carry out timing analysis only in that area and to do that we perform what is known as incremental static timing analysis it will not perform static timing analysis in the complete design but in a small portion or logic cone of our design where the transformation has impacted the time. And after we have done the incremental STA timing analysis then we say that check whether the timing target is met whether the timing target is met meaning that after because of the applying the transformations the timing might have improved and now we are meeting the target if it is meeting the target then we exit that loop of timing optimization this is the loop of timing optimization we exit the loop of the timing optimization if the timing target is met or if no possible improvement is possible may be that we have put some limit on the that how much improvement how many times of how many trials of transformation we want to apply during timing driven optimization or some criteria to check whether improvement is possible in our design or not if that criteria is met meaning that in note now we are we now condition is met which says that the no further or not significant improvement of timing is expected by doing this kind of transformation if that criteria is met then again this loop is exited . Now we carry out these transformations iteratively and if these two targets are sorry for these two conditions meaning that either the timing target is met or no further optimization is possible if that is it then it exits the timing optimization loop and it goes into what is known as area recovery.

So, once we do timing driven optimization what happens is that the timing of our design changes and it may happen that in some portion of our design the timing profile has improved such that now there are scope of or the slack has become so positive more example may that slack may become say plus 1000 just an example . So, if the slack is too highly positive then there is some scope of may be may be improving the area why if there can be scope of improving the area the reason is that what happens is that typically whenever area is decreased area is decreased meaning that area is improved the timing degrades timing or delay goes on increasing . So delay and area those goes in the opposite direction . So, if there is a if there is a the timing target if in a portion of a logic cone the timing profile has improved such that the slack is highly slack is highly positive it means that we can we can afford to increase the delay we can afford to increase the delay and as such we can also improve the area we can minimize the area as a result of that the if the delay increases slightly we still have sufficient margin that timing timing target will not be not will be become bad or not will get be affected . So, after we have done the loop of timing driven optimization a phase of area recovery is is carried out in which we try to improve the area in the region of the circuit where we can afford to increase the delay and once the area recovery phase is also done then we then the tool will the implementation tool will produce the timing optimized mapped netlist.

So, this is the basic flow of how the timing driven optimization is done. Now we will be looking into what are the various kinds of transformations that can be applied in timing driven optimization. So, we have seen in the earlier lectures that libraries contain cells of same functionality but of different sizes for example, an inverter can be of size 1 x where x is some arbitrary unit of area and it can have a cell of size say 2 x and 4 x and so on similarly for other types of logic gates. So, the library contains cells of the same functionality and different sizes . Now the size of the cell increases then what happens. suppose we are considering this inverter and 1 x inverter and 2 x inverter then what will happen in 2 x inverter the transistors that are used to implement the inverter that will be of larger size .

So, that the w by l width will be larger for the transistors the width will be larger for the transistors in the implementation 2 x compared to 1 x and it will be further larger in 4 x . So, the width of the transistor becomes larger as a result what happens is that the size increases and the size of the cell increases and the other thing is that the delay decreases. Now why does the delay decrease? In this case the reason is that if w by l of a transistor increases then the current that is driven by that transistor increases . So, current is proportional to w by l and if w by l increases the current increases and therefore, if this inverter suppose this inverter was charged was connecting or driving a load of value c l then because of the bigger transistor in 2 x compared to 1 x this 2 x 2 x inverter will charge this capacitor faster and therefore, its delay will be less compared to 1 x and in 4 x the delay will be further reduced. Now in resizing what we do is that in resizing we change the size of the cell, but keep the function of the cell the same .

So, we replace an AND gate with an AND gate, but of say larger or smaller size. So, that is what resizing is. So, in resizing we replace a cell c 1 with another cell c 2 that produces the same Boolean function, but has a different size ok. So, let us take an example. So, let us assume that there is a cell c 1. Let us assume for the sake of illustration that c 1 is an inverter .

So, suppose this one is an inverter, this is c 1 and what we are saying is that we replaced it with functionally equivalent cell c 2 of a larger size. So, now, we replaced it with a bigger inverter which is say c 2 . This is what the resizing is: we change the size of the cell. Now because of changing the size, what effects do we expect? Suppose this cell was driving another cell say d 1 .

So, we are changing the size of c 1 to the size of c 1, but not of d 1 . So, here also we will have a d 1 d 1. Now because of because of larger size of c 2 the and because c 2 is comprising of bigger transistor larger transistor it will be a able to drive this load corresponding to this wire this wire and the input capacitance of this inverter d 1 there will be some load capacitance . Now if we consider this timing arc the delay of this timing arc will be smaller for c 2 compared to c 1 . So, delay and output slew of c 2 can be reduced .

So, here the delay between the arc between let us call this pin as a and this as z. So, between a and z the delay will decrease because c 2 is of larger size and therefore, it will drive the or it will drive the load quickly and the delay will decrease and the other important thing is that its output slew will also decrease. So, if it was rising slowly the output when we replace it by c 2 it will rise sharply . So, the output slew will also be decreased in the case of a big if we use c 2 of a larger size and as such if the output slew decreases what happens to the delay of the transistor d 1. Now in the earlier lectures we have seen that the delay is a function of output load and the input slew.

Now when we increase the size of c 2 what have increase the size of this this cell we use a bigger cell c 2 then what happens that the output slew decreases and therefore, we expect that the d 1 delay of d 1 will also come down why because the input slew has has become lower compared to if there was c 1 instead of c 2 . So, because of this replacing c 1 with c 2 of larger size we expect that the delay and output slew of c 2 will reduce and the delay of cells in the fan out of c 2 can also reduce meaning that delay of d 1 can also reduce . And there will be other effects on the input side. This is the good effect in timing or in delay that will happen in the c c 2 and in its fan out , but what will happen in the fan in of that that we need to consider also. Now suppose this c 1 was being driven by an inverter of size of whose name let us take the name as say b 1 . Now we are not changing the size of b 1.

So, b 1 will also be driving c 2 this b 1 . Now what impact of resizing of c 1 will be there of that will be seen by b 1 . Now what will happen at b 1 is that earlier it was driving a cell c 1 which is of smaller size now it has to drive a cell c 2 which is of larger size and therefore, the input capacitance of the pin a in this case compared to this case will be lower . So, if we make a bigger cell in that is we use a bigger cell c 2 the input pin capacitance of a will be larger and therefore, b 1 will need to drive a larger load . And therefore, what we expect we expect that the delay of b 1 will go up delay of b 1 will go up why because delay is a function of load and input slew since the load has increased on b 1 the the the delay of b 1 will increase and also the output slew of b 1 will increase the output slew suppose here the the b 1 signal was rising like the output at this point b 1 was rising like this at after we have resize its slew will increase and it will rise at a slower rate .

Now what impact this increased output slew can have is that it can actually increase the delay of c 1 or c 2 . So, the delay and output slew of the driver of c 2 can increase and the delay of the cells in the fan out including that of c 2 suppose there was another gate in the fan out let us call it as e 1   here also you have e 1 . Now the delay of e 1 will also get impacted because the output slew of b 1 has increased. So, the delay of e 1 can also increase . So, delay of the cells in its fan out including that of c 2 can increase .

So, what this is example illustrates is that if we increase the size of cell c 1 the delay can in decrease in c 1 and in its fan out, but delay can increase in its fan in and because of the output slew it the delay of the resize cell can also increase and in the fan out of the driver of of the the resize cell there also the the the delay can increase. So, these two effects are in opposite directions: the first effect is actually decreasing the delay and the second effect is increasing the delay. Now if we resize we have to pick an optimum . So, if we go on increasing the size of c 1 at some point the second effect will be over taking the first effect and we may not get the required benefit of resizing . So, increasing the cell size will not always yield to improvement in delay .

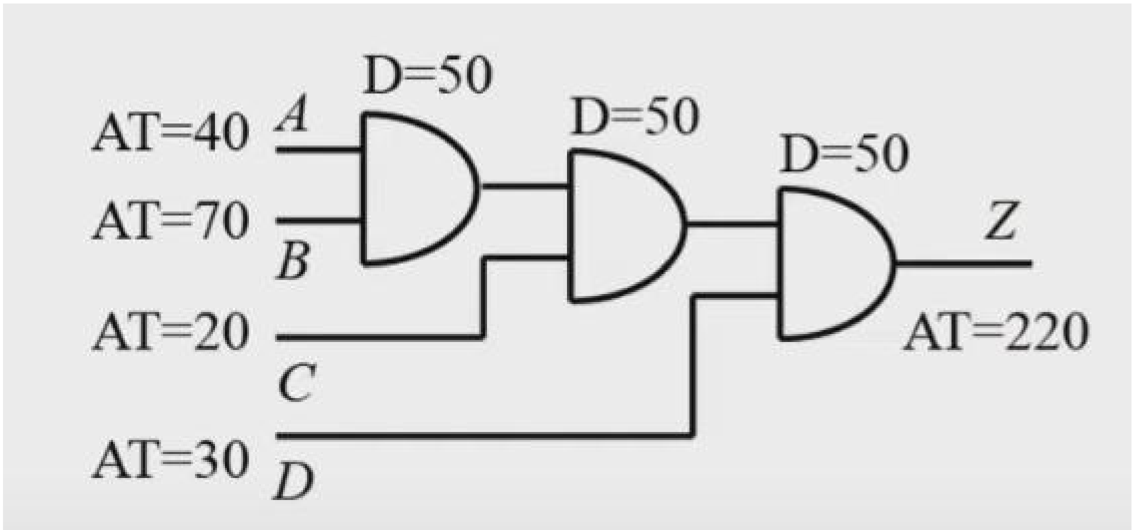
So, we have the tools needed to do and do or pick the size of the cells more intelligent .

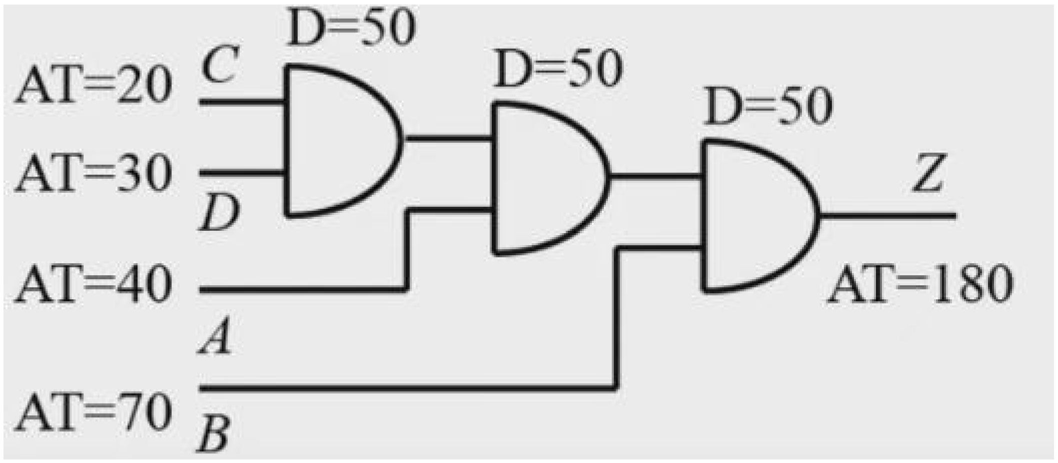
So, the tools for tools use some algorithm to find an optimum size of the cells and use it and use it for timing driven optimization. Now let us look into another optimization technique which is known as restructuring. So, in restructuring or rewiring what we do is that we in the timing critical signals are moved closer to the sinks in the cone of logic to reduce the overall part. So, let us take an example and understand what we mean by moving a critical signal close to the sink .

Consider this circuit . So, there are 3 AND gates and the arrival time of the signal at a is 40, at b is 70, at c is 20 and b is 30. Now out of these 4 signals which one is most timing critical the one which is most timing critical is this which is coming at b why because the arrival time is maximum for this . So, now what is the worst arrival time at this point z or the maximum arrival time. So, the maximum arrival time is if we consider this path the delay of all the AND gates. We have assumed it to be say 50, 50 time units and we have taken time units as arbitrary numbers . So, suppose we consider the path a to z.

So, what is the delay from a to z 50 plus 50 plus 50 150 plus 40, 40 is the arrival time. So, the arrival time at this point through a will be 190 through b how much it will be 150 plus 70 that is 220 . Now out of 190 and 220, 220 is the worst for others there are only 2 AND gates. So, 50 plus 50 plus 20 is 120 and 50 plus 30 is 80. So, for others who are not critical the critical path is this .

So, the critical or the worst path is this . So, we will want to move b closer to the sink. So, the sink here is z where the signal is ending . So, how can we do that? So, we can just do the rewiring . So, we can put the signal which is coming at at b directly to this gate .

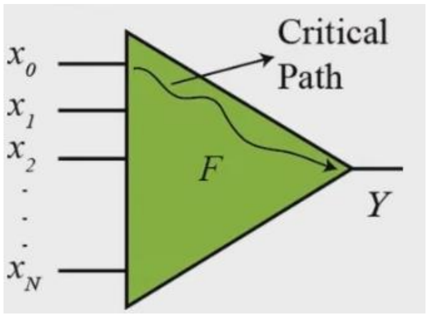




Since this is all of them are AND gates . So, whichever signal is applied at which of the ends does not matter and therefore, we can move the signal which was coming at b closer to the to the sink . And then the one which is having the next arrival time is 40 and then 30 then 20. So, now let us see the arrival time. So, for this path what is the arrival time? 20 plus 150 that is 170 for the other path from this we have 180 50 30 plus 150 that is 180.

So, the worst is 180. Now for this path we have 100 plus 40 that is 140. So, still 180 is the worst and for this path we have 50 plus 70, 120. So, the worst is now from c sorry from this this. So, the worst path which is exhibiting the worst arrival time or the maximum arrival time is this sorry is this which is exhibiting an arrival time of 180. So, from 220 we have improved the arrival time to 180 by rewiring.

So, this one is a simple case because all these logic gates are of the same time AND gate and therefore, if we move any of them it does not matter because at this end the function is symmetric . So, this is just a 4 input AND gate . So, that is what so therefore, we can put any signal at any of the end the functionality is not going to change. But what if this box that is shown here consists of having any arbitrary function ? If it was any arbitrary function then how can we restructure let us see.



So let us suppose that y is a y y y is the output produced by a combinational logic function f and it depends on the signals x 0 x 1 x 2 to x n .

$$Y = F(x_0, x_1, x_2, ..., x_n)$$

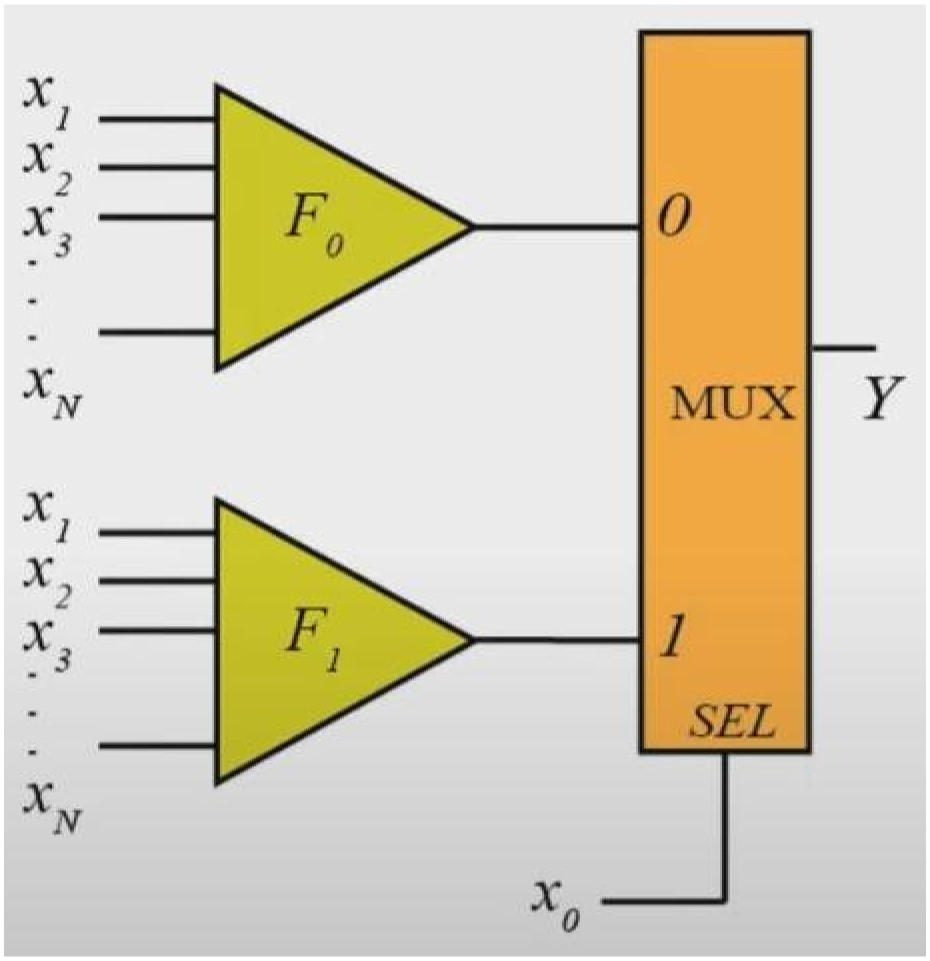
So, these are the inputs and it is a combinational logic and let us assume that let us assume that the x naught is critical suppose whatever the the most critical are out of it sorry out of all the inputs x 0 to x n the one which is most critical let us call it as x n and this is the path which is most critical and f is an arbitrary combinational function . So, y is equal to f x naught x 1 x 2 to x n . Now what if x naught is critical what do we want to do we want to move x naught closer to the output y we want to move closer to the output. How can we do it? So, let us see. So, the problem is how to restructure the circuit such that the worst delay of the circuit improves so the arrival time at this point y improves that is what our target is .

So, what we can do is that we can do Shannon expansion whatever this function is if we can do a Shannon expansion with respect to x naught . So, to do Shannon expansion to do Shannon expansion what we have seen earlier is that we need to put the value of the variable as 0 if we put x naught as 0 then f we get a function f naught which is the negative cofactor . So, this is a negative cofactor and then we put the value of x naught as 1 and we get f 1 and f 1 is known as positive cofactor . Now, once we have negative cofactor and positive cofactor we can implement this function in terms of multiplexer and this cofactors how we can implement it like this.

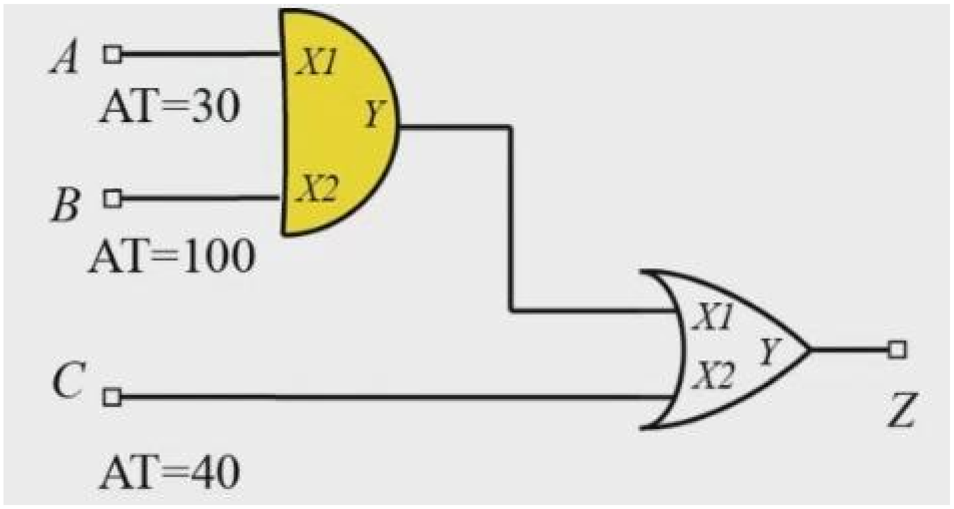
$$F_0 = F(0, x_1, x_2, ..., x_n)$$

$$F_1 = F(1, x_1, x_2, ..., x_n)$$

So, we use a 2 to 1 multiplexer. So, this is a 2 to 1 multiplexer in which in the select line we have x naught on the select line we have x naught. So, whenever x naught takes a value of 0 this f naught should be produced. So, now we implement f naught and if the select x naught is taking a value of 1 then we select f 1 or the positive cofactor . And the combined result of this selection is that what y will be y will be nothing, but y is equal to x naught bar f naught plus x 1 sorry x 0 f . So, this is what this implementation is implementing .



Now, let us take an example, then this concept will become more clear. Suppose we were given this circuit. This circuit is the combinational logic circuit in which we have an AND gate and an OR gate . Let us assume that the delay of the AND gate is 25 in some arbitrary timing units and the delay of OR gate is 20 . And the delay of the multiplexer is 30. We will be using a multiplexer later on.



So, the delay is specified . Now, let us look into the arrival time profile. So, arrival time at A is 30, arrival time at B is 100 and arrival time C is 40. So, at the point z what is the worst or maximum arrival time? First let us take this path. In this path we have 2 1 1 AND gate 25 plus 20 that is 45 plus 30. So, this is the time to arrive. So, on this path the arrival time will be 75. Through the other path we have 100 plus 20 plus 25 that is 145.

$$Z = F(A,B,C) = AB + C$$

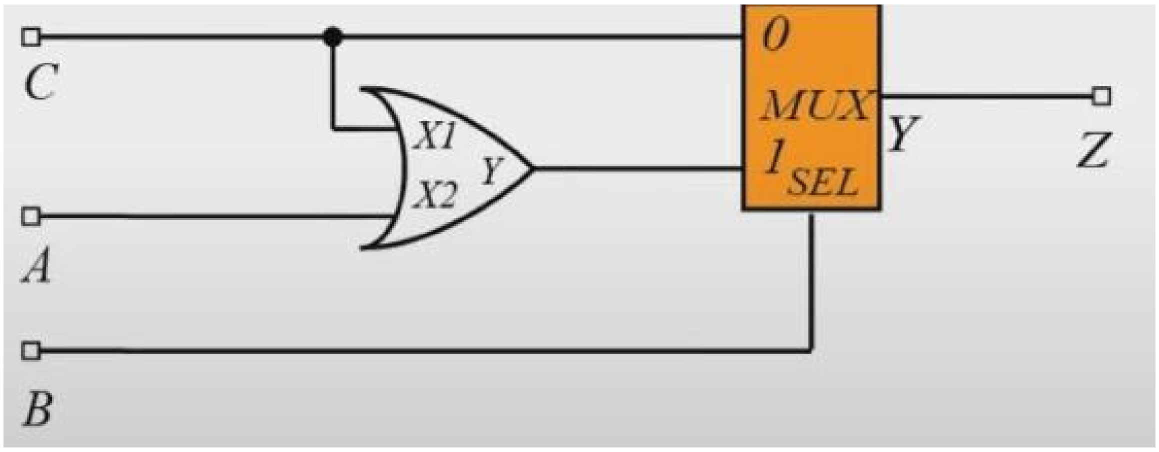
And for the third path we have 40 plus 20 that is 60. So, out of this the worst is through the B . So, the worst the maximum arrival time is through this and the and the arrival time is 140 . Now, can we reduce this arrival time by making some transformation in the logic that is the purpose of this restructuring for timing driven optimization. So, to improve the arrival time at the node at the output z we need to move this to the base signal B closer to the output .

And therefore, we need to do a Shannon expansion of this circuit way or the logic function with respect to the function B. So, first let us see what z is implementing. So, z is implementing A B . So, this is A B.

So, this is A B and this is an OR gate. So, we have A B plus. Now we want to do a Shannon expansion of this function with respect to B. Why with respect to B because this path through B was the most timing critic . We want to improve the timing of that. So, what will be the positive negative cofactor of this for that we need to set B is equal to 0 .

$$F_{B=0} = C \quad F_{B=1} = A + C$$

So, if we put B is equal to 0 z becomes C . And if we make B equal to 1 we get the positive cofactor . So, if we make B equal to 1 then become A plus C . So, that is now with this cofactoring now we can do the implementation using mux. And so the implementation will look something like this. So, now the signal B is going to the select input of the mux .



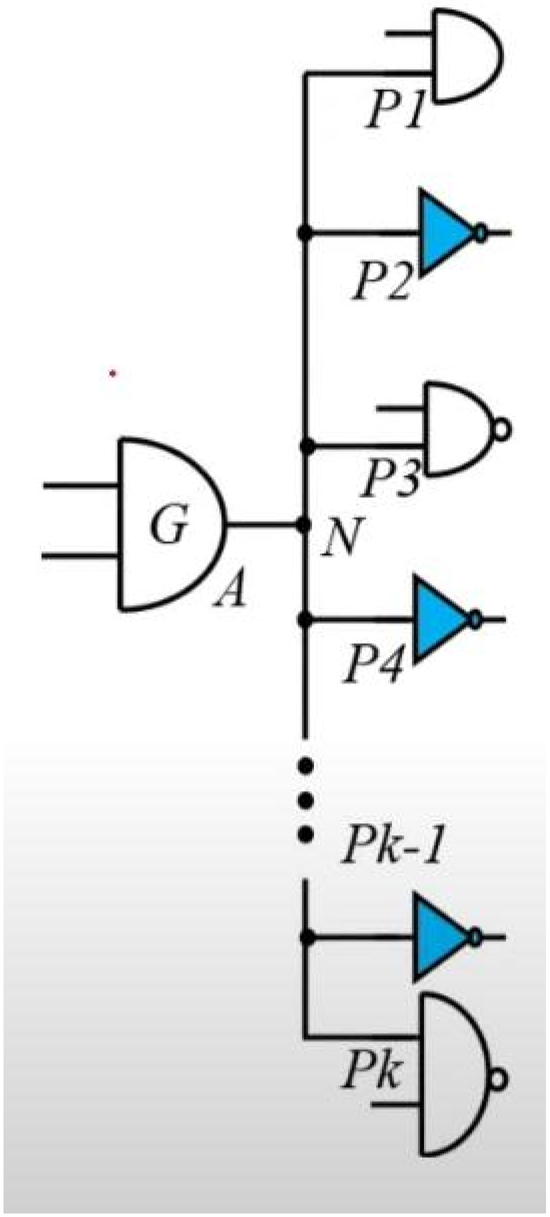
And on the 0 side we have the negative cofactor that is C. And on the 1 1 pin we have the other cofactor that is A plus A plus C . The positive cofactor that is A plus C implemented using this OR gate. Now let us understand and evaluate the arrival time. So, the arrival time at C is 40. It is already given here .

B is 100 and A is 30 . Now if we consider this path what is the arrival time? The arrival time is 40 plus 30 is 70 . If we consider this path again 30 sorry for this path A is equal to 1. So, this is the arrival time.

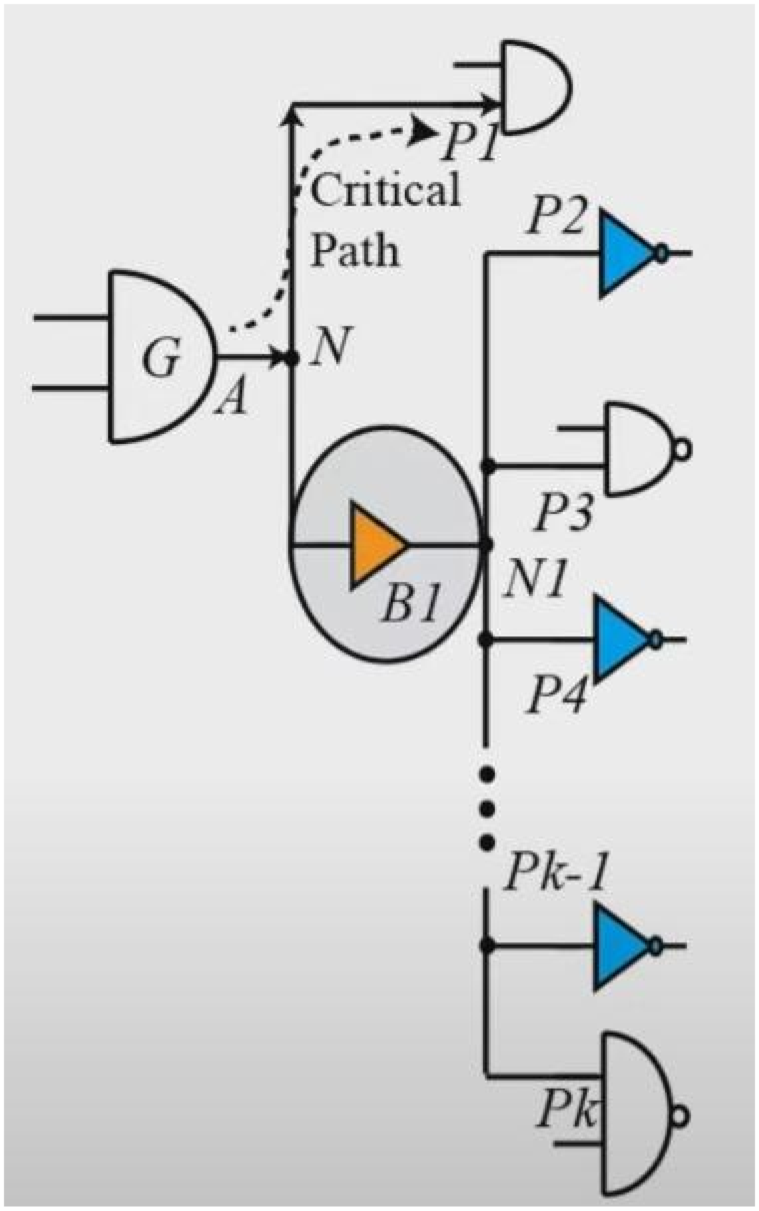
We have to take this one . So, 40 plus 20 plus 30. So, 30 so the arrival time will be 40 plus 30 is 70 plus 20 that is 90 . Then we have to see the other path . The other path is through this 30 plus 20 plus 30.

So, this is the arrival time. So, 30 is equal to 80. And the third one is from this to this and this 100 plus 30 is 130 . So, the worst arrival or the maximum arrival time is 130 . So, if what we have done is that we have moved the late arriving signal that is the B closer to the output by doing a factoring using Shannon expansion theorem . And as a result the arrival time or worst arrival time decreased from 145 to 130 and therefore we have improved the timing of our design . Now what will be the cost involved? The cost involved can be in terms of area .

So, it may happen that F 0 and F 1 which are the cofactors if we implement them then there can be the the number of logic elements in implementing F 0 and F 1 may be larger than the original circuit and therefore there will be an area penalty or the area of the circuit can increase in this kind of transformation. Similarly we have used say there are if there are say n input inputs in this logic cone   here it was only 3 inputs A B C if there are multiple inputs then we can do it recursively   if we pick the first one or the last arriving signal and do a Shannon expansion with respect to that and then we pick the next late arriving signal and do the transformation with respect to or carry out Shannon expansion with respect to that variable and so on and we can do this recursively. Now let us look into another optimization which is known as fan out optimization. So, in fan out optimization what we do is that we insert buffers in a high fan out net to improve the timing or other other characteristics of our design. Suppose this was a circuit so there is a gate G in which the output is driving many logic gates .

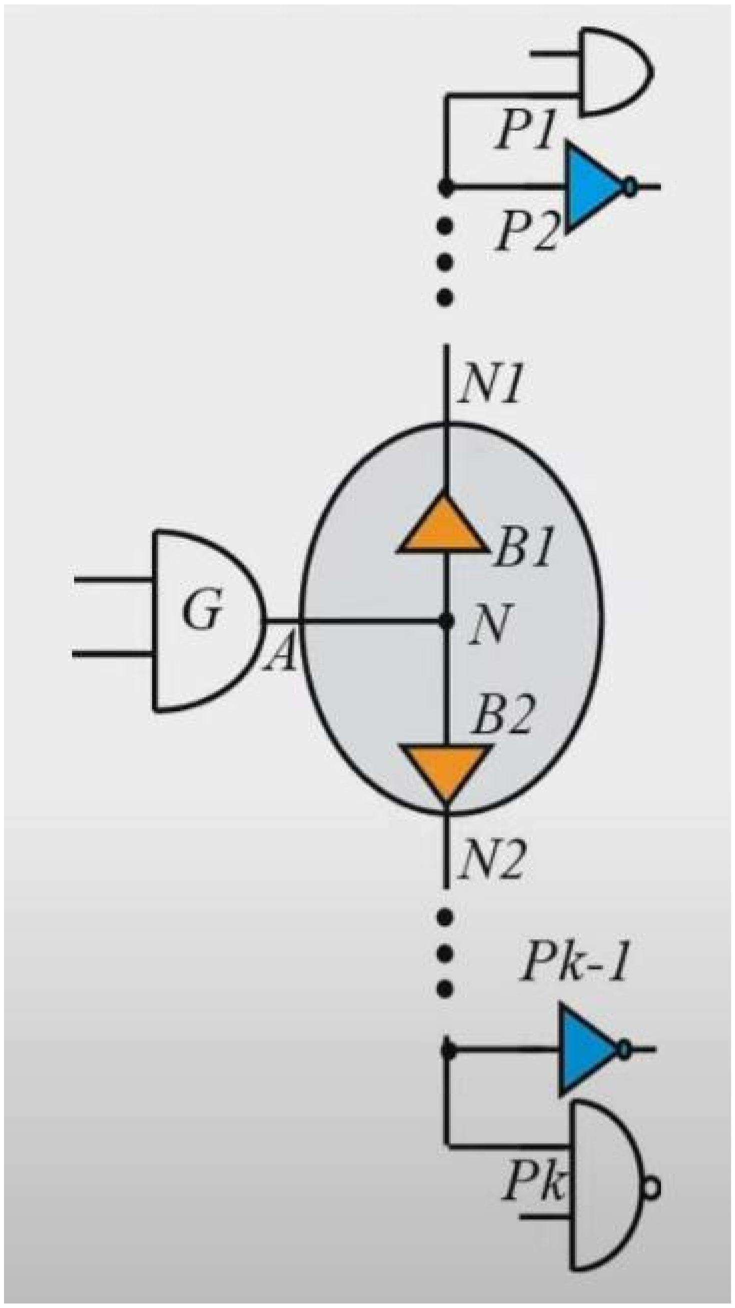


So, suppose there were K K pins which were driven by this logic gate G. If G sorry if K is large then the load that is being driven by this gate can be much higher and therefore, the delay of G may be very large and we may want to reduce the delay of G to then how we can reduce the delay. Suppose out of all the things that is p 1 p 2 p 3 p 4 to p K the p 1 was most critical suppose this path was most critical. So, what we can do is that we can add a buffer for the other rest of the sinks . So, we add a buffer B 1 which is driving p 2 p 3 to p K and the p 1 which is timing critical that is directly driven by this gate .



So, now as a result of what this transformation what will happen the the low load that is load on G gate will reduce why because earlier the all this loads were seen by the gate G there is all these loads p 1 p 2 to p K was seen by the load by the gate G and therefore, the output capacitance that was being driven by G was much higher . Now what happens is that the output capacitance that is driven by G will be lower because now the capacitance that will be seen by G will be because of this p 1 the pin p 1 and the B 1 that is it . The other loads will be shielded by the or that will be driven indirectly by B 1 and therefore, the delay of G is expected to decrease and the and the timing of the critical path is expected to improve . Now if there are more signals which are more critical then we can add more buffers in these. So, now suppose B 1 is still suffering from a large delay and there are some paths for example, p 2 is also next critical. If p 2 is also critical then we can again shield the rest of the dry sinks from B 1 by doing a next level of buffering and so on.

So, what are the downsides of this transformation? The downside is of course, the area taken by the buffer B 1 and also the path the delay of the pass p 2 p 3 p 4 to p K the delay of the path going through these pins will encounter the delay of the buffer B 1 also and therefore, the timing of the other paths that is through p 2 to p K that can actually degrade . So, that is another thing that we need to take into account . Now this high finite net optimization can also be done to fix the load violations and also slew violations.

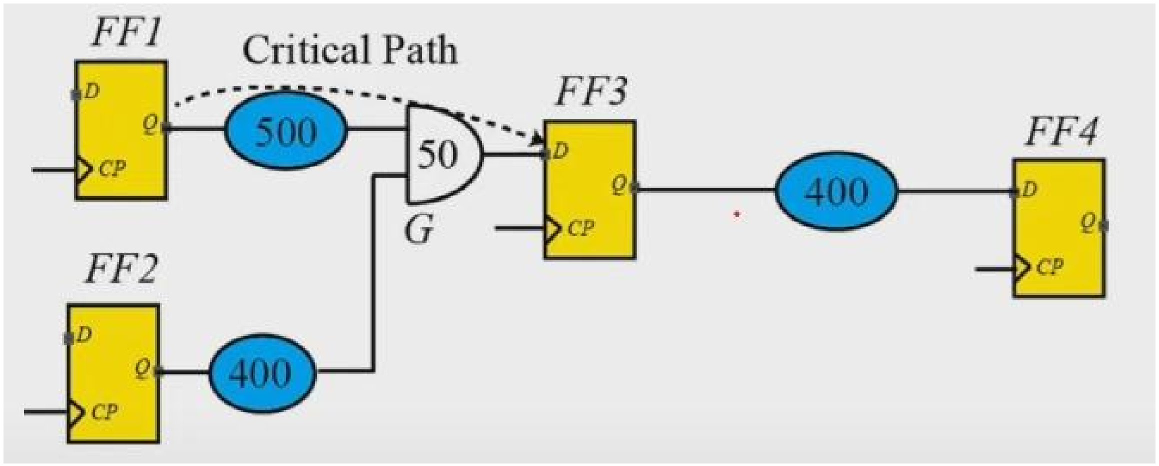


For example, assume that the output slew of the gate G was very high, it was rising very very slowly such that the max transition violation was there on this net or if the load that was driven by G was so high that the max capacitance violation was encountered by G. Now in that case what to do? In that case we can add buffers to this net to the net A or this net N and as a result of that the slew and output slew violation and the max capacitance violation can be fixed.

What can we do? Let us see. So, we can add two buffers B 1 and B 2 . Now suppose there were say 100 100 100 sink pins p 1 p 2 p 3 to p 100 suppose there were 100 100 pins in the fan out . So, what we can do is that we can add two buffers B 1 which is driving 50 p 1 to p 50 and B 2 is driving 51 to p 51 sorry p 51 top 100. So, we have distributed the fan out with two buffers. Now again if this load is high for example p 1 to p 50 is still high we can further divide by adding more buffers .

So, we can divide it into two more buffers. So, here we will add two more buffers each driving a load of say 25 sinks and so on. So, this can be done recursively. Now again here what penalty we are incurring the penalty is in terms of area . So, there will be area and also power dissipation associated with the new buffers that we are adding in our design.

Now let us look into one more transformation which is known as retiming. So, what do retiming do? A retiming balances the amount of logic between registers. So, let us take an example then it will be clear. Suppose there is a portion of our design which is the critical portion . So, in this assumption, for the sake of analysis let us assume that the clock skew is 0 and assume that the ideal clocks are coming at all in all the clock pins .



So, in that case the setup constraint will be decided by the data path . So, in this case the data path delay is say 500 plus 50 from this path from this FF1 to FF3 500 plus 50. So, it is 550. The other path is 400 plus 50 so 450 and this path is 400 this is the delay of this path is 400.

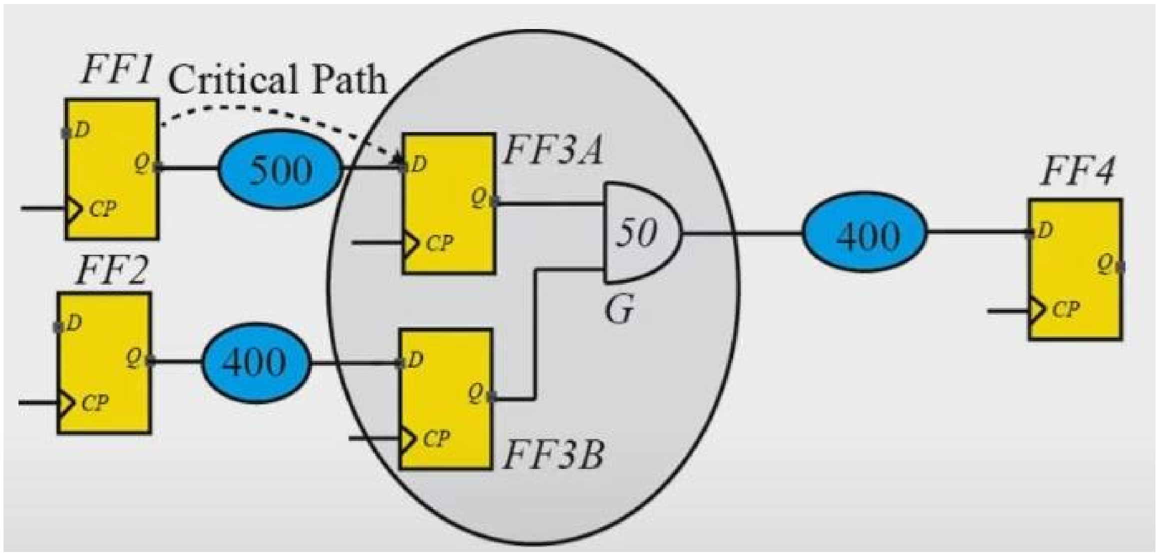
Fmax = 1/550ps

So, these are the delays . Now out of 450 550 and 400 the maximum is 550. So, the clock frequency with which this circuit will work will depend on the critical path . So, the critical path is this and its delay is 550. So, if the clock skew is assumed to be 0 and the clock to skew delays are assumed to be 0 setup time is assumed to be 0 just for the sake of illustration let us assume them to be 0. Then what maximum clock frequency the circuit can operate the maximum clock frequency will be t is equal to or the minimum clock period will be t is equal to 550.

Let us assume that the unit of time is picosecond. So, it is equal to 550 picosecond and the maximum frequency at which this circuit can operate is 1 by 550 picosecond and that will give us a clock frequency . Now the thing is that it can reduce this delay of the critical path. If we can reduce the delay of the critical path then we can reduce it then we can make the time period further smaller and therefore, we can increase the frequency of our design . So, how can we decrease the delay of the critical path? So, we can decrease the delay by noticing that on the other side of the flip flop FF3 we have a delay of only 400 .

So, can we do a transformation in the circuit such that some computational load that is being done on the critical path is done on the other path because on the other path there is some margin we have because here we have a delay of 550 . On the other side of the flip flop we have a delay of 400 . So, there is some margin left on the other side. Can we shift some computation burden from this path on one side of FF3 to the other side. If we can do that then we will be able to decrease the delay of the critical path and increase the clock frequency.

So, what is this transformation and how can we do this transformation? So, this can be done using this . So, what we have done is that we have moved this AND gate to the other side of the flip flop, but since the AND gate is taking 2 inputs we will have to replicate this in this flip flop FF3. So, we have FF3 a I mean FF3 b and the output of this is fed to this AND gate and then that goes to the FF3 . So, functionally this circuit and this circuit are equivalent. There is no difference in the functionality because whatever the output was here at these points let us call it as x and let us call it as y and this is z .



So, z was equal to x and y. In the other circuit what we have done is that we have taken x and y directly at this at this flip flops and in the next clock cycle this will be x and this one will be y and this point will be x and y and that will go to the D pin of FF4 and therefore, this circuit is is functionally equivalent to the other circuit. Now what about the delay or the or the delay of the critical path. Now delay of the critical path in this case is for this path we have 50 only sorry 500 only because the AND gate has moved to the other side on the for this it is same 400 and on the other side the delay has actually increased from 400 to 450 yet the maximum delay is 500 this is 450 this is 500 and this is 400. So, out of these numbers the one which is maximum is 500 and that will be still critical and the maximum clock frequency will be decided by that. Now in this circuit the time period must be greater than 500 picosecond while in this circuit the initial circuit To must be greater than 550 picosecond.

So, here the frequency f1 is equal to maximum frequency is 1550 and here the maximum frequency is 1 by 500 and therefore, we have increased the operable frequency in this circuit. But what penalty we did we paid we paid the penalty of extra flip flops earlier we had the earlier we had only one flip flop now we have two flip flops. So, this kind of transformation is known as retiming. So, the transformations that we discussed are only some of the transformations that are done by the logic synthesis tool during timing driven optimizations. There are many other types of transformations also done by the tool which we have not discussed.

So, if you want to look into more such transformation you can refer to this book. Now to summarize in this lecture what we have done is that we have looked into an important aspect of logic synthesis that is timing driven optimization and we also discussed a few important transformations that are done in timing driven optimization. So, after we have done logic and timing driven optimization our timing of our circuit has improved. But what about other aspects of our circuit for example, power . So, we need to also do power analysis and power optimization in our design flow.

So, in the next lecture we will be looking into power analysis and power optimization. Thank you very much.