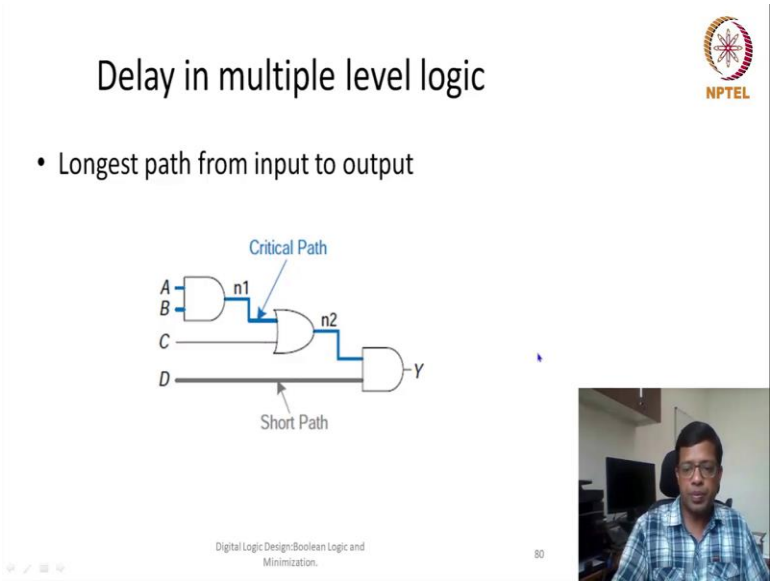
Digital System Design
Professor. Neeraj Goel
Department of Computer Science Engineering
Indian Institute of Technology, Ropar
Multi-level logic

(Refer Slide Time: 0:16)



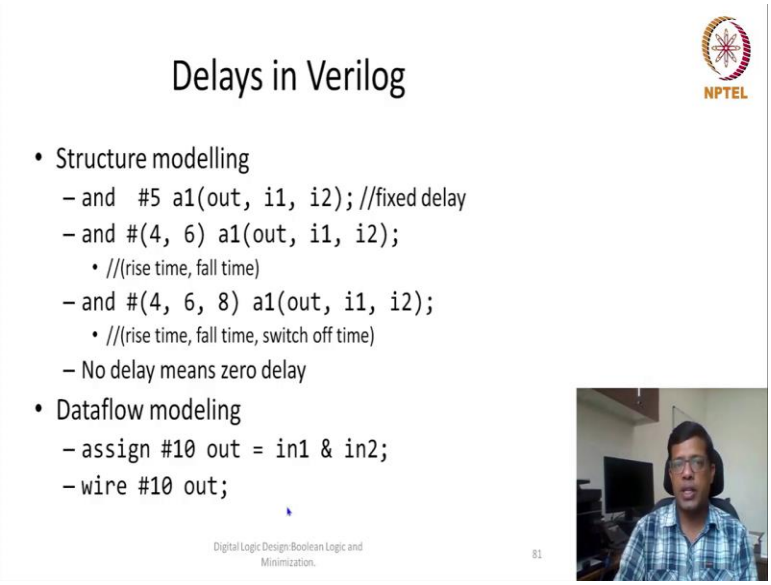
So, if the gate is of multiple stages or multiple levels, then what is the delay? Delay would depend on the longest path from input to output. Again the assumption is that all the inputs are available at same time, if all the inputs are will change at same time or are available at same time, then we can find out what is the longest path. So, this longest path is also called critical path, this critical path would define the delay. So, it even, in two logic level implementation this particular definition can works, so whatever is the longest path.

So, in two level logic optimization if at first level a particular gate has more number of input, so that will form the critical path, that would be part of critical path. So, here we see that let us say this is the logic implementation we have done. So, in this logic implementation, this is the shortest path because, whenever this D input is available, after that there is only one gate delay to change the output. However, if any of these inputs A or B will change and first n1 will change, then n2 will change, then Y will change.

So, this become the critical path and again as I said before, so this if we have different paths, so some paths are short path, some are much larger in delay. So, that would also means that in this

Y output, there will be more number of glitches, this Y would keep on transitioning from 0 to 1, 1 to 0 before it will get stabilized to the final output.

(Refer Slide Time: 2:06)



So, with let us let us see that now, if these delays are there, and we are using Verilog for modeling of this gate level implementation, if we want to represent these delays in in very long implementation, how do we do? So, there are two kinds of modeling we have studied in our previous lecture of Verilog. So, we have seen structural modeling and data flow modeling in structural modeling, the modeling is done using basic gates AND gate, OR gate, NOT gate, if we insert, if we tell that what is the delay of that particular AND gate, OR gate or NOT gate that is sufficient.

So, if the delay of all the basic gates are known, then whatever is built using that gate, so let us say in our previous assignment, we have designed XOR gate. So, from the XOR gate, let us say we even create something else, that delay would be imparted, delay would be introduced in the XOR gate and the same delay could be propagated. So, the other logic which would be implemented using that XOR gate will also have those delay characteristics.

So, what I mean to say here, I would rephrase this as that if I want to implement, if I want to model delay in my structural modeling, in my leaf level design wherever AND gate, OR gate, NOR gate or basic gates are instantiated, then I should write my delay statements there I should tell that what would be the delay of this particular AND gate. So, here I wanted to say that delay

of my AND gate is 5 units. You remember that this AND is the AND is the gate declaration and this is the instance name, a1 is the a1 instance of my AND gate. And this hash 5 essentially means that whenever input i1 and i2 will change after 5 minutes of time my out will change.

This delay is also written as some time two values like in the parentheses we write two values. So, first value represent the rise time, the second value represent the fall time. So, rise time is the output, if output is changing from 0 to 1, then this would be the delay; if output is changing from 1 to 0 then this would be the delay. So, as I said, initially in the starting of this lecture that my output delay change, it depends on the transition whether the transition is from 1 to 0 or 0 to 1. So, that particular thing can also be modeled using Verilog.

There could be third parameter to delay also that is rise time, fall time and switch off time. So, in switch off time is a special like which we have not considered so far. So, if delay is, if the output is neither high nor low but it is at a high impedance or it is disconnected or it was switched off. So, from switch of how much time it will take to go into a state of 0 or state or 1 that is called switch off time.

So, in most of our implementation to make it simple, we only consider one delay but if we want to model more number of things, then these delays are also considered. So, this is when out of these three input, one of the input is not specified, only two inputs are specified, it automatically take it as a rise time and fall time and only one input is specified that means the delay is fixed for all the cases. So, it essentially means that for rise time also delay would be 5, for fall time also delay would be 5 or switch off time also delay would be 5.

In case of there is no specification of hash, it means that there is no delay or we can also call it 0 delay. In case of data flow modeling, there are two ways of specifying delay; one that whenever we do assign, whenever there is assign statement, after assign statement we can write hash 10 or hash whatever time. So, after that much amount of time this input, change would be reflected in the output. So, this means that this is also an inertial delay or propagation delay that whenever input changes would be there the output would change after that much amount of time.

So, there is one small caveat here, that if my input changes, let us say input change that 0 time and after that input again changed at time 8, then there is a possibility that there is no transition at output because to make it certain, although we said in hardware things are uncertain, but Verilog

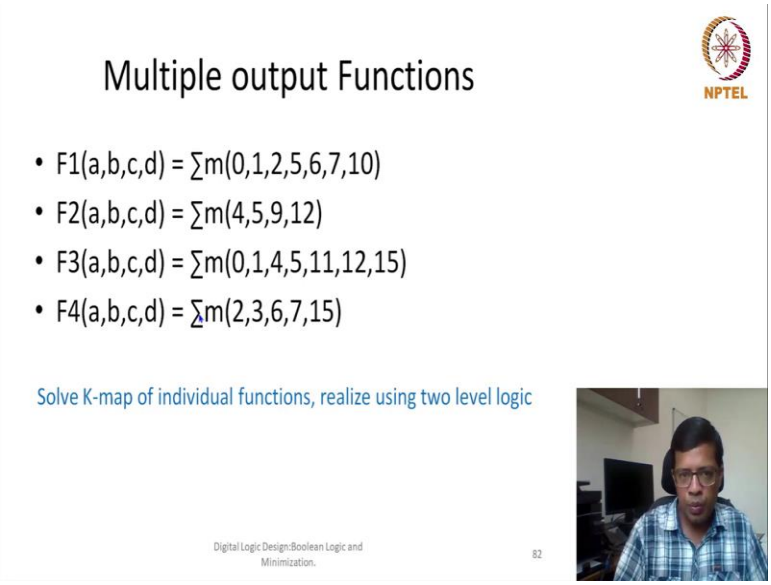
being a deterministic language here we have a very clear cut rules that if there is another change in the input, then this hash 10 may not apply. So, because my input has further changed before the delay of this particular transition, so it would depend on the what is the impact of this after, before this this completion of delay of this transition.

So, in other words, we have to say that to make it correct, my input delay should, my input should not change before that delay of, before the delay of the transition or before the propagation delay. And there is also a possibility of propagation delay because of wires. So, we can say here we can also define propagation delay on wires. So, that means, whenever we define this out as a wire, we can also say hash 10, which means that whenever anything is assigned to out there would be a delay of 10. So, these two statements are slightly different. In the first statement, we are saying that this expression would be computed and that computation would be given after a certain time to my output.

So, this expression we can use for delay in the, in calculating any kind of operations. Here it is a logical AND operation. So, logical AND, OR, NOT or XOR, all those operations delay we can we can use this using this data flow modeling, but there are some delays because of just a wires. So, there could be simple assigned statement, let us say in1 equal to out. So, if we use this particular statement that any assignment to out will again lead to a delay of 10. So, sometimes this is required because some wires will have propagation delay because those wires are our long wires and because although voltage and current can reach at instantaneous time, but in VLSI or in this these chips, the chips are also very, very small, incredibly small.

If the distance is in the terms of nanometer or hundreds of nanometers, then also the delay of those wires could be there. So, because of that, we also consider the propagation delay of wires and that is why we will also model it. So, this is how those delays can be modeled in Verilog, so that we can see the overall functionality in terms of delay for a particular circuit. So, this was about the delay.

(Refer Slide Time: 10:31)



So, there are a couple of small topics which I would like to cover. So, so far we have seen F expression; any function, any particular expression or any particular function usually have a single input, single output. So, there are multiple inputs, but there is a single output. So, there is also an additional possibility that the same inputs, there could be multiple functions. So, here I am taking some example that A B C D are the inputs, but they are for different functions F1, F2, F3, F4, all of these functions are represented in in canonical sum of mean terms.

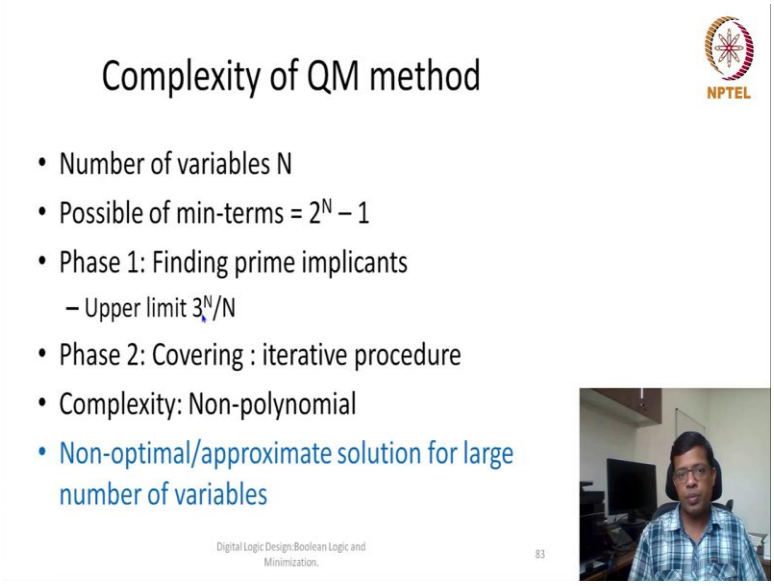
Now, when I am designing them, will there be any impact or how should I design? As far as designing is concerned or implementation is concerned, so we all these functions F1, F2, F3, F4, we would develop or we would put independent K map for all of them. And after independence solving of K maps, we will write this F1, F2, F3, F4, as a independent sum of product expressions or independent two level logic. So, even though they are multiple output, it does not matter from the implementation perspective or from the design perspective, if we are using two level implementations.

However, to save area, what we can do? Some time that, let us say some particular product term has appeared in function one as well as in function two. So, because same product term is there in function one and function two, so we can we can reuse that particular sum term, product term and if we use that product term, so that we some area could be could be minimized. So, that essentially means that, if multiple output functions are there, they give us certain opportunity to

reduce the area. So, we can find out if there is some common product terms or if there are some common sum terms, those common sum terms can be utilized to reduce the area.

So, what would be the impact there would be the fan-out of that particular gate would increase. So, let us say there is a common product which is there in this function as well as this function, so that means that product term we can instantiate once, but the output has to go to F1 as well as F2. So, the fan-out of that particular gate would increase. But effectively it is a good optimization, because we are reducing the area with a slight increase in the delay. Delay would increase because otherwise fan-out has to be 1 but now fan-out is 2.

(Refer Slide Time: 13:30)



So, however like yes, so one more small topic we would like to cover that what is the complexity of Quine–McCluskey method. So, this or optimization, what is the complexity of the Boolean optimizations, which we have done. And, so how we were doing Boolean optimization? So, first we were constructing, we, for given an expression first we were constructing in the terms of sum of min terms or some max terms that means, we were representing our function in a canonical form.

And given N variables, if the number of variables are N, then what is the possibility how many min terms or how many max terms would be there? That the maximum possibility of N terms, min terms could be 2 is to power N minus 1. So, for example, for N equal to 4, there could be 15

min terms which are possible. So, even though the total number of min terms may not be 2 is to power N, but it could be of the order of 2 is to power N.

So, let us say we have, so far we have solved multiple of the examples where number of min terms, if the number of variables are 4 is in the range of 8 9 10 11. So, that means it is of the order, it is nearby that 2 is to power N minus 1 range. Now, if we see, if we follow closely how this Quine–McCluskey method is working, so this QM method is working by comparing each min term with the other min term to find that either is a one variable difference or not.

So, that means the comparisons are also of the order of 2 is to power N, it is not only 2 is to power N, so you are, ideally you would like to compare each of this min term, 2 is to power N min term with the other 2 is to power N min terms. The total number of comparisons would be 2 is to power 2n in that case, but as we have seen in our previous lecture, we are doing grouping so that the number of comparisons could be minimized. But still, our observation was that the number of comparisons are still more than 2 is to power N.

So, whatever were the number of min terms, the number of comparisons, were still much more than that. So, this first stage of prime implicants, where the number of stages are also of the order of N minus 1, capital N minus 1. So, first we are reducing one variable, then the second variable, then the third variable. So, the number of stages are also of the order of N and in each stage, the number of comparisons are exponential. So, total number of comparisons are kind of non-polynomial.

So, some theoreticians have given the total number of or maximum number of prime implicants given, the number of variables N is equal to 3 is to power N divided by N. So, that means the total number of x prime x implicants which could be there is again is the power of N. So, and N is a variable. Similarly, in the second phase, covering is also an iterative process. So, we first have removed the essential prime implicants by saying that these are the implicants which are covered by only min terms which are covered by only few implicants or only one implicant that become essential prime implicant gone.

And after that remaining prime (implicants), remaining prime implicants we are still covering min term one by one and total number of min terms are exponential. So, the overall cost of this QM method is non-polynomial. What it means, that non-polynomial, what is the polynomial

complexity? Polynomial means your, complexity is the power of N is to power 2, N is to power 3, N is to power 4, N is to power x, where x is a degree, but if the order is the complexity is equal to 3 is to power N here, so that means, it would exponentially would increase if N will increase.

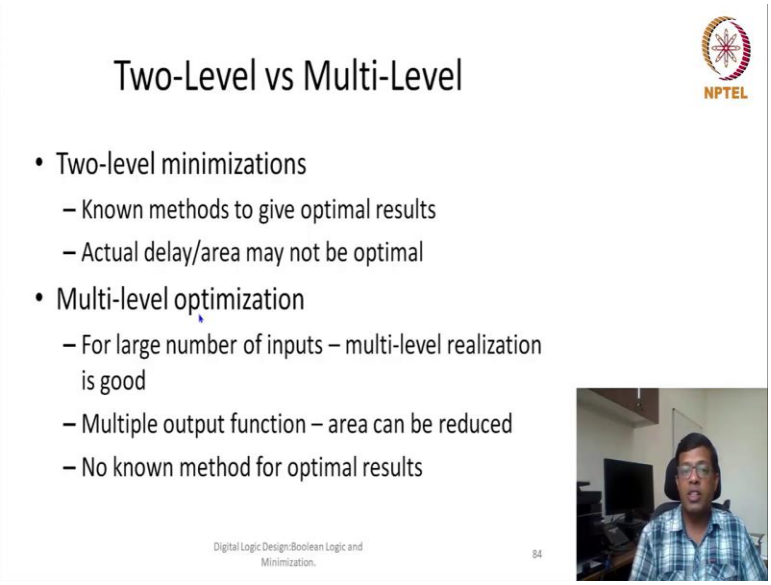
Which essentially means that if the number of variables in the logic minimization is of the order of 3 4 5 6 we would be able to find an optimal solution using K map or Quine–McCluskey method. But, if the number of variables grows beyond a certain number, let us say beyond 10 whatever computational power we have today, it will not be sufficient to find an optimal solution. If optimal solution is not there, what will we do? We will try to find a non-optimal solution or an approximate solution which is also close to optimal solution not very bad, but still okay. Why we are, why we will resort to this approximate solution because optimal solution will take exponential amount of time and that exponential amount of time is not worth.

So, let us say I have some 10 or 12 variables, because of those 10 or 12 variables my machine is occupied for whole day. And in the end, I may find an optimal solution which has 1 literal less than the non-optimal solution. It is not worth correct. So, that is why we find a trade-off, we say that non-optimal solutions are also okay if the number of variables are very, very large, because the optimal solution is exponential in nature, non-polynomial in nature and these NP type of problems, the amount of time it takes that is why we have to resort to some kind of an approximate solution.

And how do we calculate this approximate solution, this will come from the algebra. So, in the algebra we have seen that, in our previous module we have seen there were a couple of stages that we were trying to remove, redundant terms, try to add some terms and then remove some terms, try to remove variables, trying to remove literals. So, all those logic expressions could be done one by one all these logic expressions would be done one by one and iteratively, so that we reach near an optimal solution. If we reach at an, at a solution which from where we see that no further cost can be reduced, then we go, we settle out for that particular solution.

So, now, because Quine–McCluskey is not an optimal, will not be able to give optimal solution in a real time for large number of variables, we also look for other methods.

(Refer Slide Time: 21:10)



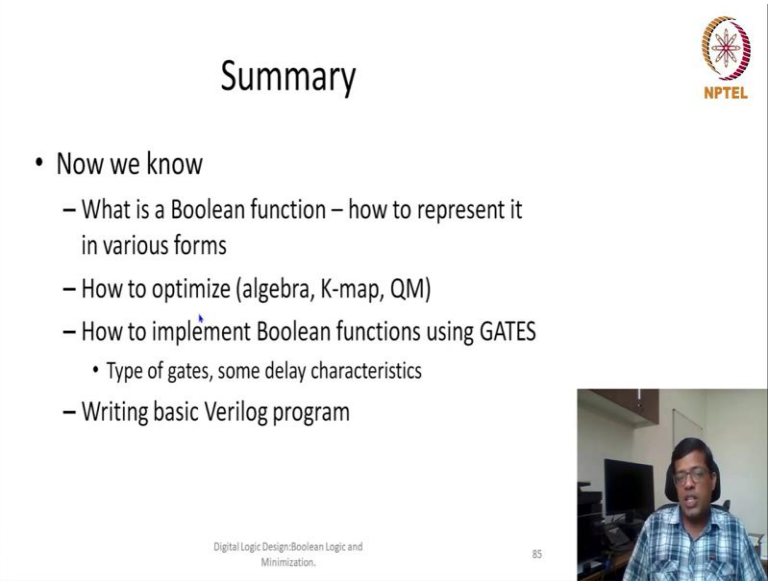
So, now, we see that, so we have seen in this particular lecture that some time we would go for a two level optimization, sometime we are going for a multiple level optimization. So, if number of inputs are large that also lead us to suggest that you should not go for two level optimization, but you should go for a multiple level optimization, because finally implementation would require a gate which has a smaller fan-in.

So, then this also gives us another question that why not to design an optimization or design a final circuit which has a multiple level optimization? Starting from the scratch, starting from whenever we started optimizing why not to go for multiple level optimization? So, the reason for that is that we do not go for multiple level optimizations because even two level optimization problem is hard, computationally hard, it is non polynomial in nature, although, known methods exist for optimal results.

In multiple level optimizations, there is no known method for even for giving optimal results. So, because there is no known method, but, so that is why keeping multiple level optimization as a goal is not usually done, multiple level optimization is done as a side effects. So, whenever we see large number of inputs, then we try to do multiple level realization and whenever multiple output function is there, then also we try to see that if there is something, some common factors are there between two functions, so that we can optimize on delay, optimize on area.

So, this multiple level optimization is usually the side effects so, we start with two level optimization, after we are done with two level optimization then to finally go for a circuit implementation, it is usually is done by software, it is done by tool itself, we do not go for manual multiple level optimizations, but yes the, for example, this Verilog compiler when it will implement the circuit, then it will go for multiple level optimization wherever it see that two level of division is not optimal or area or delay will not be optimal there, so it will break it into multiple levels and then it will also try to see what other factors can we bring brought out, so that we can further take the best advantage of multiple level optimization.

(Refer Slide Time: 24:00)



So, with this we close this particular module and I would like to summarize that in this particular module we have seen, what is a Boolean function? How do we represent it using various different forms? We can represent it using truth table, we can represent it using Boolean expressions. And we can also, we have also seen that two Boolean expression could also be equivalent. And we have also seen various laws of Boolean algebra, we have seen that these laws are slightly different from our regular algebra, but are interesting because they can help us in in much more optimizations.

After understanding these Boolean functions in Boolean algebra, then we have seen that how we can optimize a Boolean expression, how we can reduce the number of product terms, number of sum terms or number of literals? We have seen that we can use that either using algebra or using

a graphical representation like K map or a tabular representation like QM method. All of these methods were good and they were essentially using the same basic functions, the same basic criteria, same algebraic simplifications, but K map and QM are more systematic, because we know that they will give a certain optimal solution.

After that, we have also seen how these expressions or optimal expressions how they would be implemented using gates, we have seen different kinds of gates AND, OR, XOR, XNOR, NOT. So, all these all these gates we have seen what are their characteristics, how we can convert AND gate into a NAND gate, NAND gate into a AND gate and OR gate. So, those transformations we have seen. We have also seen their delay characteristics, what is the meaning of a propagation delay. Along with that, we have also seen how to write a basic Verilog program.

So, one thing which is missing in this whole story, we do not know from where these Boolean equations or Boolean expressions are coming? Who is giving you these Boolean expressions to optimize? Of course, you can say your faculty, your instructor is saying that optimize this particular expression, but how these expressions came into his mind or basically, who is the, who is creating those Boolean functions, what is the meaning of those Boolean functions? So, all this particular question, we will start answering in our next module. So, till then, thank you very much. Have a good day.