



**Digital System Design**  
**Professor. Neeraj Goel**  
**Department of Computer Science Engineering**  
**Indian Institute of Technology, Ropar**  
**Area Delay Model**

Hello everybody. Today is our last lecture of this particular module. So, in this lecture we will summarize whatever we have done in this module and we will also see how the Boolean expressions would be implemented using gates.

(Refer Slide Time: 0:38)




- Logic minimization objective
  - Cost (area and delay)
  - Implementation options
- Optimality



Digital Logic Design: Boolean Logic and Minimization. 74

And what would be the, we started our logic minimization with the objective of cost optimization. So, in this lecture we will also see what is the actual meaning of cost we have perceived, we started with some notion and we will try to elaborate that particular notion of cost. And we will also see whatever methods we have studied, what is the optimality of those methods or is there anything better or worse.


(Refer Slide Time: 1:17)



## Logic optimization

- Input : Boolean expression
- Output: Optimal two level logic (either SOP or POS)
- Optimality criteria
  - Minimum number of product/sum terms
  - Minimum number of literals in each product/sum term

Digital Logic Design: Boolean Logic and Minimization. 75



So, then we started logic optimization. We considered that a Boolean expression is given as an input and also over the last couple of lectures we have seen that the same Boolean expression or the same Boolean function could be expressed in form of multiple functions or multiple expressions. While all these expressions could be equivalent because they would generate finally same truth table. With this particular idea, since there are multiple options, multiple expressions for the same function, so the optimality idea came into picture.


So, we would like to represent the whole function in a expression which has a minimal hardware cost. And how have we defined the optimality criteria, how we had defined that a particular expression will lead to a less cost? The cost function in our previous lecture, we have treated like two points; one that the number of product terms in a sum of products, number of product terms should be minimum and two expressions with same number of products then the number of literals in each product should be minimum.

So, with these criteria we can see that whether which expression is less costly or more optimal. Similarly, for us product of sums, the number of sum terms should be minimum and the number of literals in each sum term should also be minimum if the number of sum terms are same. So, let us and in this optimality criteria, we have further assumed that the input is available as both, in original form as well as in the complemented form. So, let us try to understand what is the

meaning of this optimality criteria are in terms of hardware implementation or implementation using logic level gates.

So, to understand this we have to see that each product term in sum of product expression, each product term would lead to one particular AND gate and the number of literals in that product term would be the number of input in that AND gate. And when we are doing sum of products, so these all these product terms would finally be added, so there would be an OR gate in the end and the input of, number of inputs to that OR gate would be the, would be equal to number of product terms. So, that means if we are able to understand what is the impact of input, number of inputs or number of outputs of a gate, then we would be able to understand what would be these optimality criteria.

(Refer Slide Time: 4:34)




### Area and Delay

- Area depends on number of inputs
- Delay
  - Input / Fan-in
  - Output / Fan-out

$$T_p = a_1 F_{in} + a_2 F_{in}^2 + a_3 F_{out}$$

- Transistor size is usually modified for large fan-in gates



Digital Logic Design: Boolean Logic and Minimization. 76

Now when we talk about hardware, in hardware the cost is a vague term, it could lead to multiple of these things, multiple of matrices. So, let us say two important matrices for hardware cost is called area and delay. What is area? How much silicon or how much area, how much silicon we are going to consume when we implement a particular expression. And delay is end to end delay, so let us say our inputs are available when the output is available that would tell us what is the delay.

Now given a particular gate, area depends on, area depends on how many gates and how many inputs are there in each gate, these two things would be sufficient to know what would be the

area. Along with that it also depends on how many (interconnections), how many wires are there, so those also add to the overall area, but in general the number of gates give us a sufficient idea that what would be the area of particular chip.

Now the second part is delay. So, if we know the delay of each gate then we can also calculate, we can also find out the delay of whole expression or whole logic whatever we are implementing. So, for one particular gate, the delay depends on how many inputs are there and the output is going to how many other gates. So, there is also equivalent terms for number of inputs of the gate, it is called fan-in. Fan-in means how many inputs are there and fan-out means a particular output is going to how many other gates as input.

So, a typical expression says that your delay, propagation delay, delay of a particular gate depends on almost square of fan-in and it linearly depends on number of outputs. As a generic expression, we can also write it that propagation delay is equal to  $a_1 F_{in}$ ,  $F_{in}$  is the number of Fan-in, number of Fan-in or number of inputs plus  $a_2$  into  $F_{in}^2$  or number of input square plus  $a_2$  into  $F_{out}$  plus  $a_3$  into  $F_{out}$ .

So, this expression essentially means that, my delay would linearly depends on my fan-out or to how many other gates my output is going, it depends squarely proportional to number of inputs. So, it essentially means that if my fan-in, the number of input or gate is 4, then with respect to there is another gate which, another let us say AND gate which has 8 inputs, so the delay of 8 input AND gate would be much more than the 4 input AND gates.

So, this expression, this linear expression or this linear expression which we written for propagation delay is sort of an approximate expression and  $a_1$ ,  $a_2$ ,  $a_3$ , these constants are also depends on so many other things like what technology or how you are designing these gates, further because people know that, designer who design all these gates, they know that input have a severe impact on the delay. So, they also do some things in the design to turn around that.

So, for example, they change the internal design. How length, breadth of the transistor is designed, so they change that length and width parameters so that delay does not increase super linearly. They try to make impact of this fan-in as minimum as possible but that would also give us side effects in terms of much more increase in the area. So, for now we can say that yes, there is an impact of fan-in and fan-out. Also there are some other factors which would depend on like

which would help in calculating area and delay that would be like technology or how designers are designing.

So, this information is essentially sufficient for now that delay would increase more than linearly, linearly or squarely proportional to fan-in and also as the fan-out will increase, then also delay would increase. And so there is also an impact of area, so number of inputs would certainly increase the area but because transistor sizing get effected as the large fan-in in the gate would be there, number of inputs increases. So, this size sometimes also increases not linearly but super linearly in case of large fan-in gates. So, let us now understand what this particular delay, what is the meaning of this propagation delay.

(Refer Slide Time: 10:22)

**Propagation delay**

Delay of GATE depends on

- Type of GATE
- Input transition type (0->1 or 1->0)
- Output transition type
- Fan in and Fan-out

Maximum Gate delay can be used For analysis

Propagation Delay

Digital Logic Design: Boolean Logic and Minimization. 77

The slide features the NPTEL logo in the top right corner and a small video inset of a presenter in the bottom right corner. The timing diagram shows two input signals rising simultaneously, followed by the output signal rising after a delay labeled  $T_p$ .

So, to understand the propagation delay, let us consider that we have a simple AND gate, let us say these are the inputs, two inputs and one output. We assume, let us assume that the input is changing, both these inputs are changing at same time. Now the delay or propagation delay essentially means that the output would change, output will not change instantaneously but it will change after some time. So, this whole diagram is called timing diagram where time is the, our x-axis is a time axis and y-axis because there all of them are binary gates, so all these gates are taking binary input and output.

So, each input there would be two levels; one level represent 0, another level would represent 1. So, in this timing diagram we can understand that whenever input is changing, output would

change after some time. So, whatever is this time, this time, the difference when output is available or output changed after our change in input is called propagation delay. So, this propagation means that there would be an, there would be some slowness, whatever, whenever input is changing, output will not change instantaneously. Why it will not change instantaneously?

So, we can say there is some, on a very high level we can understand that this sort of inertia. So, we also called it inertial delay that whatever things are there in rest they would like to be in rest, so it will take some time for change the output level from 0 to 1 and 1 to 0. And if we want to go in little more detail, so it depends on different technologies, let us say these gates are implemented using CMOS technology. In CMOS, the output and input both are represented by voltage.

So, output would change from 0 to 1, so basically whenever output is 1, this 1 voltage, this high voltage is actually stored in a capacitor. And capacitor is a slow device, whenever a capacitor, you want to discharge a capacitor from 1 to 0, it would be slowly discharged. And it depends on the resistance and the capacitance that RL constant, that how much time it will take to get discharged. So, that is the actual reason of this propagation delay, it is also called inertial delay.

One more thing here that output will start changing after some time, so let us say there would be some uncertainty for this period, so like in this period some time it is possible that output goes from 1 0 0 1 but it gets stabilized after some time. So, this uncertainty is also there during some periods, so we say the latency or a propagation delay is the maximum time, after that output will not change given the input has stabilized.


If input has stabilized, an output should not change after that time. So, that is the propagation delay of a particular gate. Similarly, we can define the propagation delay of a combination of gates. So, the overall, this propagation delay as we have seen in the previous slide also, propagation delay it depends on like type of gate. So, let us say when I just say AND gate, OR gate or XOR gate, NAND gate, NOR gate. So, type of gate would certainly define what would be the propagation delay, partly because propagation delay also depends on what is your truth table.

So, practically it also happens that the delay for 1 to 0, or 0 to 1 is different. So, let us say my input transition is from 1 to 0 or 0 to 1, so both these input transition will lead to different delays. Similarly, at output also, as I said this in CMOS technology it depends on the capacitor, so 1 to 0 would require a different delay rise time and 0 to 1, 0 to 1 would require a different delay rise time from 1 to 0 the fault time could be different.

So, this 1 to 0, 0 to 1, this transition may cause different delays. So, and of course we said that for example, this AND gate, if the number of input to this AND gate changes, then also delay will change. If this output instead of going to one particular gate, it also goes to 10 other different gates, then also a delay maybe different. So, with all this uncertainty, how do we model delay and how do we optimize at logic level or in this (( ))(15:35), we do not know about fan-in and fan-out, about these technologies factors. So, what do we do? We take certain assumptions, many a times we take the maximum gate delay for analysis.

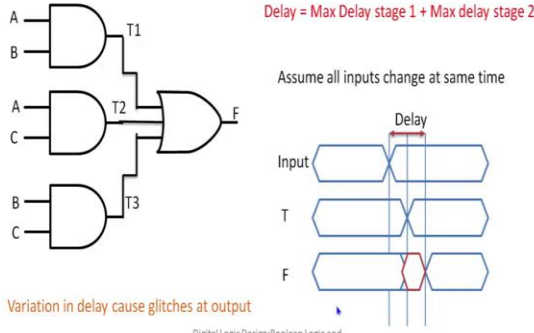
So, let us say, 0 to 1 and 1 to 0, they are two different delays for the output, so whatever is the maximum we can take that particular delay as the propagation delay of that particular gate. So, other than that, so let us say if different fan-ins are there, because fan-in has more contribution factor, so we can say there could be different models given for fan-in of the gate, so we can say that two input AND gate has so much delay, three input AND gate has so much delay. So, these kind of things would help us in simplifying these facts, otherwise life at this logic level estimation or logic level optimization would be much more difficult.

(Refer Slide Time: 16:39)



### Delay of two level logic

- For example:  $F = AB + AC + BC$




$\text{Delay} = \text{Max Delay stage 1} + \text{Max delay stage 2}$

Assume all inputs change at same time

Variation in delay cause glitches at output

Digital Logic Design: Boolean Logic and Minimization.



So, given this, let us see that our standard two level logic, which we has been optimizing for so far. What would be the delay of that, so that we can see how our optimization will work. So, let us say we take an example, that we have this sum of product expression, so we see that for each product A B, A C and B C, the number of literals in each product term will dictate how many inputs to AND gate would be there. Similarly, the number of product terms will tell how many inputs would be there for the OR gate.

So, in other words, number of product terms will define that how many AND gates are there and how many literals in each product term will define what is the input. So, grossly we can say that if number of product terms are minimized, so that means number of AND gates are minimized, also number of input may OR gate would be minimized. Further, if I have two expressions which has same number of products terms, but in one the product term the number of inputs are less or number of literals are less, that means that particular AND gate will have less number of inputs.

So, that means delay of that particular AND gate would be less. So, effectively whatever we have defined at a very high level would serve most of the purpose that would define the delay of a two level logic. So, if we say, whatever optimality function we have considered that would also say that the delay of that particular logic would be lesser than known optimal one. So, if I want to see the timing diagram of this circuit, let me assume that all the inputs are changing, will change at the same time.



So, because the inputs are multiple, so we will define it like a hexagon, so some let us say A B C, so some of the input could be high, some of the input could be less. And at a particular time let us say this particular time, inputs are changing so because input will not change instantaneously as output will not change instantaneously, this output will also be input to something. So, that will also not change instantaneously, so there would be some kind of a slant in this line. So, 0 to 1 there would be some delay.

So, let us take this, whenever the values are at 0.5, let us consider the delay from that particular point. Now inputs are changing, after this inputs, after some time, this T signals will change. And this T signals means start changing at earlier point but let us say some T1, T1 can change little early depends on the number of inputs of these gates. Here input of fan-in of all the three gates are same. Fan-out of all them is also same, so we can assume the delay of all the three AND gates would be same. So, T1, T2, T3 will change almost at the same time.

But in other expressions, let us say a particular one AND gate has 4 inputs, another AND gate has 3 inputs or 2 inputs, so the gate which has 4 inputs would give the output after little more time than the gate with a smaller input, smaller fan-in. So, after all of these inputs has changed, then at the second level F will change after some more delay. The total delay would be the sum of two levels, so there would be delay at first level, all of these things are computed in parallel because all the inputs are available at same time, all the outputs T1, T2, T3 would be available at almost same time.

So, that delay I can say the delay at first level and similarly delay at the second level means when these all of these outputs are available then my OR gate will start computing and F would be available after another set of delay. This whole delay would be the propagation delay. Now from this method, we can also see that if my logic is of two level, means only some of products or product of some that also means that my delay would be optimal. So, if number of logic levels will increase, that will also increase the delay of a particular expression.

So, overall if I want to calculate the delay in this two stages, I can also say the delay would be the maximum delay of stage one because only after that, let us say one AND gate has a 4 input or 5 inputs, only after that particular gate has computed, the expression, the output of that AND gate then only OR, final output of the OR gate will start changing. So, this would depend on

maximum delay of stage one and the maximum delay of stage two, that would define the net total delay.

The other thing also we have to note is that, because these inputs maybe available at different time  $T_1$ ,  $T_2$ ,  $T_3$ , so there would be certain time here in the output when output would be fluctuating. So let us consider that  $T_1$ , this particular AND gate has 4 inputs, all other has 2 inputs, so because  $T_2$  and  $T_3$  are available early, so there would be, they will change the value of  $F$ . So there would be certain value of  $F$ .


After that,  $T_1$  will come after some time and  $T_1$  will further change the output of, the value of  $F$ , so there could be certain transitions, we also some time called it there would be spikes, there would be some time this  $F$  would be 1 and 0, but it will get stabilized when this whole delay would be there. So, maximum delay of stage one plus maximum delay of stage two, after this much delay there will not be any change in the output.


This also, so these spikes of 1 and 0 is also called glitches at the output. So, this also give us one particular lesson, that whatever is the delay of my whole expression, here let us say delay of my expression is equal to maximum delay of stage one plus maximum delay of stage two, so what would happen if my input will change before this whole propagation delay of my expression? It would lead to uncertainty. That means we will never know what, whether the output is corresponding to the previous change or a new change.

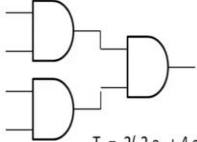
So, that is why it is always desirable, it is always, not only desirable, it is important that input should not change. If you want our output for a particular expression, function or gate; my input should not change before this whole propagation delay or because before the delay of this whole function. If input will change, that will cause uncertainty, we do not know what that output is, whether output is corresponding to previous input or a new input. So, this was about the delay.

(Refer Slide Time: 25:19)

**Impact of large fan-in**




  
 $T_p = 4 a_1 + 16 a_2 + a_3$

  
 $T_p = 2(2 a_1 + 4 a_2 + a_3)$

$T_p = a_1 F_{in} + a_2 F_{in}^2 + a_3 F_{out}$

For large N, square term dominate =>  
Conversion to Multi-level is required to minimise delays

Digital Logic Design: Boolean Logic and Minimization. 79



Now let us also consider the effect of fan-in, let us say we have a gate which has 4 inputs AND gate. Now this 4 input AND gate, we know that, if we this AND gate distribution low applies over the AND gate, so this 4 input AND gate can also be written as, we can, let us say this is A B C D and so we can say A and B could be ended, B C and D could be ended and finally the output of this AND gate can also be ended. So, these two expressions, these two logic expression or these two gate implications are actually equivalent.

And further if we say, we keep this particular equation in front of us that propagation delay depends on is the square function of my input and linear function of my output, so because here we assume the output is for both of them is same, so here it would depend on the square function of my input. If that is the case, let us take fan-in as 4, so I can say this propagation delay here would be 4 into  $a_1$  plus 16 into  $a_2$  plus  $a_3$ . And here it would be twice, because this is first level, this is second level.

So, and because both of these gates are looking equivalent, so we are simply saying that whatever is the delay of this we multiply it by 2. And delay of 1 2 input gate is 2 into  $2 a_1$  plus  $4 a_2$  plus  $a_3$ . So, we can see from this expression that even for the very small value of  $a_2$ , propagation delay of this particular gate is lesser than a single stage gate. So, what does it mean? It means that if my number of inputs are large, it always, it seems that it is better that if I break that particular gate into smaller gates with multiple stages.

How many stages would be there? So, this multiple levels or multiple stages would be there. How many stages would be there, let us say if I want to create stages with all the 2 input gates, the number of stages is going to be  $\log_2 n$  base to the number of inputs. So, let us say if the number of inputs are 8, then three stages would be sufficient. In first stage, number of gates would be 4 and then the next stage there will be 2 gates and the final stage there will be 1 stage.

So, this way in three stage, we would be able to create that particular implementation of 8 input AND gate. And delay wise it is going to be much more efficient. Yes, certainly there could be more area in this particular implementation than this implementation, so that is a trade-off that if we are focusing on delay, then this multiple stage implementation could be good. So, this gives us slightly conflicting vibes, conflicting remarks.

So, in our first remark we said that two level implementation looks like a good implementation because the delay we know is certain that it is a two stage delay, delay of first stage plus delay of second stage. Now we are saying that if there is a gate with large fan-in, then we are forcing that particular gate to be implemented as multiple stage or multiple level implementation for that particular gate.

So, what it is done finally, it depends on, what is done finally it depends on how many inputs are there and some time because if inputs are exceeding certain number then it is, it usually get implemented as multiple stage instead of two stage. Although, it would look like two stage, but finally in the hardware we will implement as multiple stage gates. And we will see in couple of minutes that there would be other advantages also if we implement it as a multiple stage gate.