

**Switching Circuits and Logic Design**  
**Prof. Indranil Sengupta**  
**Department of Computer Science and Engineering**  
**Indian Institution of Technology, Kharagpur**

**Lecture – 21**  
**Design of Adders (Part – I)**

If you recall, what we have to discussed so far; we had looked at the different ways of realizing a given function in the form of an expression some of products, product of sums. And we talked about several methods using which you can minimize the size of the expression so that the resulting gates level realization becomes cheaper in some respect; number of gates can be smaller the size of gates can also be smaller.

Now, let us look at some real circuits and how you can design those circuits and apply them in various applications. So, we shall be looking at the design of such basic building blocks which can be used to build larger systems.

To start with, we look at the most fundamental thing that we need in many many applications. Think of an adder, we need to add numbers in almost any applications you can think of. So, we shall be starting by talking about the design of adders and after that you should be looking into some other very commonly used functional blocks ok.

(Refer Slide Time: 01:47)

**Introduction**

- We shall be discussing the design of various commonly used combinational circuit modules.
- What is a combinational circuit?
  - Whose output depends only on the inputs that have been applied, and have no bearing on the past history of the inputs.
  - Common examples:
    - Adders and subtractors
    - Multiplexer ✓
    - Decoder / Demultiplexer ✓
    - Encoder, etc. ✓

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Switching Circuits & Logic Design

So, the title of this lecture is Design of Adders. So, before we move into the design of adders. So, as I had said we shall be discussing over the next few lectures, the design of various commonly used so called combinational circuit modules.

So, let us first try to understand what is a combinational circuit. See the definition of a combinational circuit is like this. Suppose I have a circuit where there are some inputs, there are some outputs. Now, if this circuit is such that the outputs will depend only on the inputs that I have applied at that point in time and they have no bearing on the past history of the inputs.

Past history what happened earlier, what inputs were applied earlier that will not have any bearing on for the output is coming. So, such a circuit is called a combinational circuit, but the output will depend only on only on what inputs we are applying at this particular point in time.

So, you think of an adder, an edition. I apply 2 inputs to be added, I get the sum as the output. This is an example of a combinational circuit ok. We talked about a multiplexer we have the inputs where the select lines, we select one of the inputs and that go to the output that is also combinational circuit. But you think of an application like this. Well you have seen the traffic light controller in the intersection of the roads.

See the sequence in which the lights glow that is not arbitrary. Red, Orange, Green then again orange then red again orange then green there is a particular sequence. So, when the lamp is orange; so, what will be the next state? What will be the next lamp that will glowing that will depend on the past history. If previously red was glowing, then the next will be green and if previously green was glowing, then the next will be red. This is an example of a so called sequential circuit which we shall be discuss in later where the output that we get depends not only on what we have now, but also some previous history of the system ok.

So, far here we are talking only about combinational circuits ok. Some of the very common examples of combinational circuits that we shall be discussing are; first we shall be talking about Adders and subtractors. Multiplexer we already have talked about earlier we shall be looking into it again, Decoders and Demultiplexers and Encoders. These are the some of the function blocks that we shall be going into some detail.

(Refer Slide Time: 05:08)

### Addition of Two Binary Digits (Bits)

- When two bits A and B are added, a sum (S) and carry (C) are generated as per the following truth table:

Inputs		Outputs	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$\begin{array}{r} 0+0=00 \\ 0+1=01 \\ 1+0=01 \\ 1+1=10 \end{array}$$

**HALF ADDER**

$S = A \oplus B$   
 $C = A \cdot B$

$S = A'B + AB'$   
 $C = A \cdot B$

$S = A \oplus B$   
 $C = A \cdot B$

$S = A \oplus B$   
 $C = A \cdot B$

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Switching Circuits & Logic Design

So, we start with addition ok. We already talked about the problem of adding binary numbers. We talked about the binary number system, we talked about how you can add to binary numbers; we start from there. From there you will get some idea that how we can design a circuit that can actually carry out addition ok. Let us see.

First we take a very simple sub problem; addition of 2 binary digits or bits. This is the truth table. So, I have 2 input bits A and B. So, I am adding them. I get a sum, I get a carry. S is the sum and C is the carry. What are the rules for addition? You have already know this. If we add 0 and 0, the sum is 0 which in binary you can write 0 0. If you add 0 and 1, sum is 1 0. If we add 1 and 0 again sum is 1 0. If you add 1 and 1 sum is 2. So, in binary it is 1 0 that is why I use 2 bits. Because in 1 bit you cannot represent the sum, 1 plus 1 is 2 1 0.

You see this truth table actually shows this. The last bit is sum 0 1 1 0 is a 0 1 1 0 and this is the carry 0 0 0 1 right. So, we want to design such a circuit and a circuit which does this is called a half adder. So, the definition of a half adder is that half adder takes 2 binary digits or bits as input A and B. It produces sum. It produces carry. And this is the truth table of a half adder.

Now directly from truth table, if you write down the expression that what is the switching function that is realizing sum in the carry. You first look at this sum. Sum has a true minterm here, it has a true minterm here. You see inputs as 0 1 and 1 0, 0 1 means A

bar B, 1 0 means  $A \bar{B}$ . You see here, we have written exactly that  $A \bar{B}$  or  $\bar{A} B$  which is nothing, but the Exclusive OR function.

So, sum is nothing, but the exclusive OR of A and B ok. If you look at the carry, carry has a single true minterm corresponds to 1 1 which is  $A B$ . So, carry is nothing, but  $A B$ . So, if you implement of such a half adder using gates, the sum you can implement by a single exclusive OR gate; the carry you can implement using a single AND gate right.

Now, one thing to notice that exclusive OR is sometimes not regarded as a basic gate; So, if you want to implement an XOR gate maybe you will be using a function like this, like you will be using 2 AND gates like this. Then there is a there will be an OR gate. Output of the AND will go to the input of the OR. First one we will get  $A \bar{B}$ . So, the input A is there; so there will be a NOT A bar and B will be fed to the second input. Second one will  $\bar{A} B$ . So, this A will be fed here and B will be fed with the NOT to here. This is the implementation of a exclusive OR function  $A \oplus B$ .

Now let us see, if we say that delta is the delay of a basic gate; we shall also talk about this later. Basic gate means AND OR and NOT. These are regarded as the basic gates. Then for generating carry, the delay is only delta. There is a single AND gate. But for some, if I realize using this you see; firstly, there is a NOT, then there is AND, then there is a OR. There are three levels.

So, the delay will be thrice delta. So, you should remember this. If delta is the delay of a basic gate, then for a half adder sum will required 3 delta and carry will require only delta to generate fine.

(Refer Slide Time: 10:11)

**Addition of Multi-bit Binary Numbers**

0010110 ← Carry	1111110 ← Carry
0101010 ← Number A	0111111 ← Number A
+ 0001000 ← Number B	+ 0000001 ← Number B
-----	-----
0110100 ← Sum S	1000000 ← Sum S

- At every bit position (stage), we require to add 3 bits:
  - 1 bit for number A ✓
  - 1 bit for number B ✓
  - 1 carry bit coming from the previous stage ✓

**WE NEED A FULL ADDER**

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Switching Circuits & Logic Design

Now from here, let us talk about how we can add multiple bits, multi bit numbers. This we have already discussed earlier. Let us look at it once more. Suppose I have a number A, I have a number B. These are the example I have taken. These are 7 bit numbers. So, I am adding 1 and 1, sum is 0 with a carry of 1. Then I am adding all these numbers with carry 1 1 and 0, sum is 0 again with the carry of 1. 1 0 0 sum is 1, but no carry 0 1 1 is 0 with the carry. 1 0 0 is 1 with no carry. 0 1 0 is 1 with no carry. 0 0 0 is here ok.

So, you see when you are actually adding 2 numbers, multi bit numbers; we also need to take care of the carry. We need to check the carry from 1 stage to the next. These are called the stages. For a 7 bit number, I say that these are 7 stages of addition. The first stage of addition generates a carry which is used in the second stage. Second stage of addition generates a carry which is used in third stage like this.

Let us take another example. So, where the numbers are like this 0 followed by all 1's and the second number is just 1. This is an example when you see that the carry will be propagating up to the last position. You see 0, we saw the initially there is no carry initially carry is 0. So, 0 1 1 will generate sum of 0 carry of 1. 1 1 0 sum 0 again carry of 1 same thing, sum 0 carry 1, sum 0 carry 1. This will continue right.

So, you see there is carry propagation from the last stage to the more significant stage. So, the reason I have taken this example is that later on we will see when we calculate the delay of such adder, will have to take care of this carry propagation time. If it is an 8

bit adder, it must propagate from stage 0 up to stage 7. We will have to take care of this entire carry propagation time.

So, one observation from here is that you see in every stage that you are shown in this example. Now we are required to add 3 bits, 2 bits are the bits of the number and the third bit is the carry which is coming from the previous page. So, 1 bit of number A, 1 bit of number B and a carry bit coming from the previous stage.

So, earlier we talked about a half adder which can add only 2 bits right. Inputs were A and B, output were sum and carry. But here, we need another kind of an adder which can add 3 bits not only A B, but also there will be a carry input 3 inputs ok. So, we need another kind of an adding block which is called a full adder. Full adder basically can add 3 bits.

(Refer Slide Time: 13:50)

**What is a Full Adder?**

- A full adder has three inputs and two outputs:
  - Inputs: two input bits A and B, the carry input  $C_{in}$ .
  - Outputs: the sum S, and the carry output  $C_{out}$ .
- To add two multi-bit numbers, we can use a cascade of full adders.

The diagram shows a central box labeled 'FA'. Three arrows point into the box from the left, labeled 'A', 'B', and ' $C_{in}$ '. Two arrows point out of the box to the right, labeled 'S' and ' $C_{out}$ '.

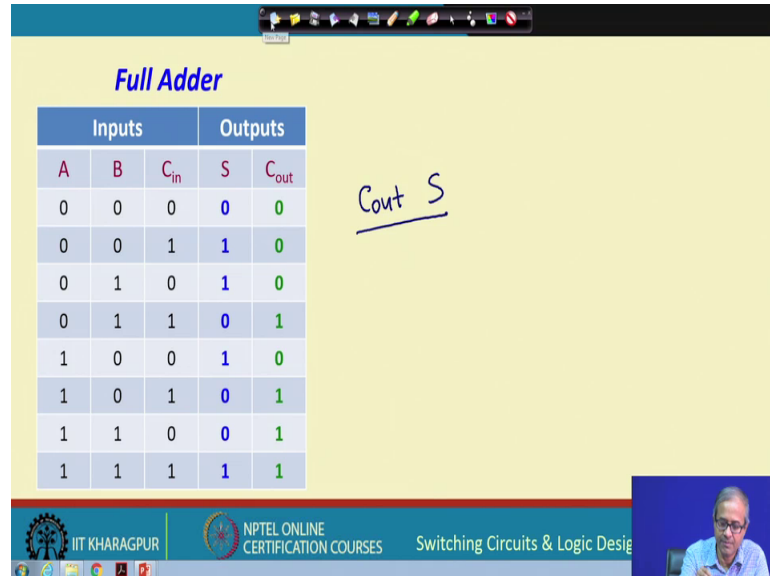
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Switching Circuits & Logic Design

Let us look at full adder. What a full adder is. Full adder as I said, will have 3 inputs. The 3 inputs are the 2 input bits A and B and the carry input let us say C in carry in and again there will be 2 outputs 1 is the sum other is the carry output. Let us call it C out.

Now, we shall see later that when you have multi bit numbers, we can use a cascade of full adders because you see exactly the way we did the addition by hand. We add 3 bits, carry was generated. Again we added 3 bits, carry was generated. Again we added 3

bits. So, if we have several full adders connected one after the other the same thing can be implemented. The idea is like this.

(Refer Slide Time: 14:57)



The image shows a presentation slide titled "Full Adder". It contains a truth table with the following data:

Inputs			Outputs	
A	B	C <sub>in</sub>	S	C <sub>out</sub>
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Handwritten note: Cout S

Footer: IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Switching Circuits & Logic Design

So, a full address schematically can be shown like this where A B and C in are the inputs and S and C out are the outputs. The truth table for the full adder will look like this. This you had this were also seen earlier. So, A B C are the 3 inputs. If we add them 0 0 0 is 0; that means, 0 0 ok. C out will be.

So, when you consider the value of the number, you consider it as C out has the most significant then S in that order. 0 0 1 is 1, 0 1. 0 1 0 sum is also 1 0 1. 0 1 1 is 2 1 0 1 0 is 2 right. 1 0 0 is 1, 0 1. 1 0 1 is 2, 1 0. 1 1 0 is also 2, 1 0. 1 1 1 is 3, 1 1 right. This is the truth table of the full order.

(Refer Slide Time: 15:45)

**Full Adder**

Inputs			Outputs	
A	B	C <sub>in</sub>	S	C <sub>out</sub>
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$S = A'B'C_{in} + A'B.C_{in}' + A.B'C_{in}' + A.B.C_{in}$$

$$= A \oplus B \oplus C_{in}$$

$$C_{out} = A'B.C_{in} + A.B'C_{in} + A.B.C_{in}' + A.B.C_{in}$$

$$= A.B + B.C_{in} + A.C_{in}$$

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Switching Circuits & Logic Design

Now, again from this truth table, you write down the expression for this sum and the carry. For this sum, you see there are 4 true minterms. If you just write them down 0 0 1 which is A bar B bar C in 0 1 0, A bar B C in bar 1 0 0, A B bar C in bar and 1 1 1. So, this again is nothing, but the exclusive OR of the 3 numbers A B and C in.

Similarly, when you look at the carry out; Here again there are 4 true minterms corresponding to 0 1 1 which is A bar B C in 1 0 1 1 1 0 and lastly 1 1 1. So, if you minimize this using Karnaugh map or any other method, you will see that the minimum form is A B plus B C in plus A C in.

So, when you implement a full adder like this using this expression, what we need is that for the first case for sum, you need either a big XOR gate with 3 inputs and for carry we will be needing 3 AND gates A B, B C, C A and an OR gate. So, each of these AND gates will having 2 inputs each. First one will be fair with A and B , next one is B and C in , B and C in and last one with A and C in like this and output will be C out. And here the inputs will be A B and C in and the output will be S.

Now, again because this XOR gate is not a basic gate; If you want implement them using AND gates, suppose this first expression we look at. Then you will need 4 bigger AND gates each with 3 inputs because there are 3 literals, each with 3 inputs. Then there will be a 4 input OR gate connecting the output of this AND gates.



And to generate this A bar B bar and C bar, there will be some NOT gates also in the output. I am not showing the exact circuit. You can draw this. So, in terms of delay again, here you see this XOR gate again will be having a delay of 3, 3 basic gates. First stage are the NOTs, then the ANDs, then the OR. But for carry out here again, there is a AND and OR; there is no NOT here too.

(Refer Slide Time: 18:57)

**Full Adder**

Inputs			Outputs	
A	B	C <sub>in</sub>	S	C <sub>out</sub>
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$S = A'B'C_{in} + A'B.C_{in}' + A.B'C_{in}' + A.B.C_{in}$$

$$= A \oplus B \oplus C_{in} \quad (3\delta)$$

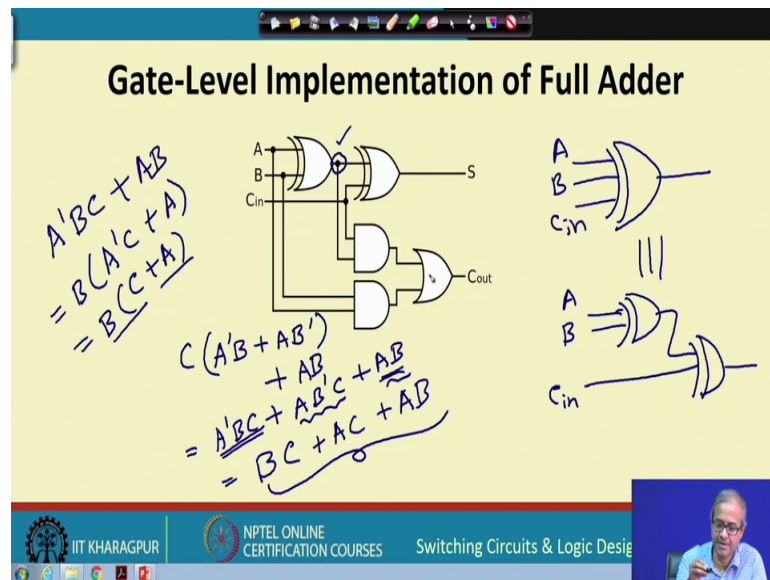
$$C_{out} = A'B.C_{in} + A.B'C_{in} + A.B.C_{in}' + A.B.C_{in}$$

$$= A.B + B.C_{in} + A.C_{in} \quad (2\delta)$$

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Switching Circuits & Logic Design

So, if you look at the delay here for a sum the delay will be 3 delta where delta is the delay of a basic gate and for carry out, it will be twice delta. Delta is the delay of a basic gate right.

(Refer Slide Time: 19:11)



There are other ways of implementing a full adder. So, I am showing a couple of such designs where you can possibly reduce the number of gates little bit. Say earlier for carry out, we had use 3 AND gates and 3 input OR gate. So, here we are making a short cut. First thing is that the sum earlier was generated by a 3 input XOR you recall.

Now, 3 input XOR gate can also be implemented as a cascade of 2 2 input XORs. If these are A B and C in, you can have A B here, you can have C in here. These two are equivalent. So, here we have done the same thing for generating sum. And for generating the carry, we are taking the output from this place. Well do not ask me how we arrived at this. This is more like an art. There is no systematic procedure for this. Somehow, someone has come up with this design which requires less number of gates and this can also generate the same carry out.

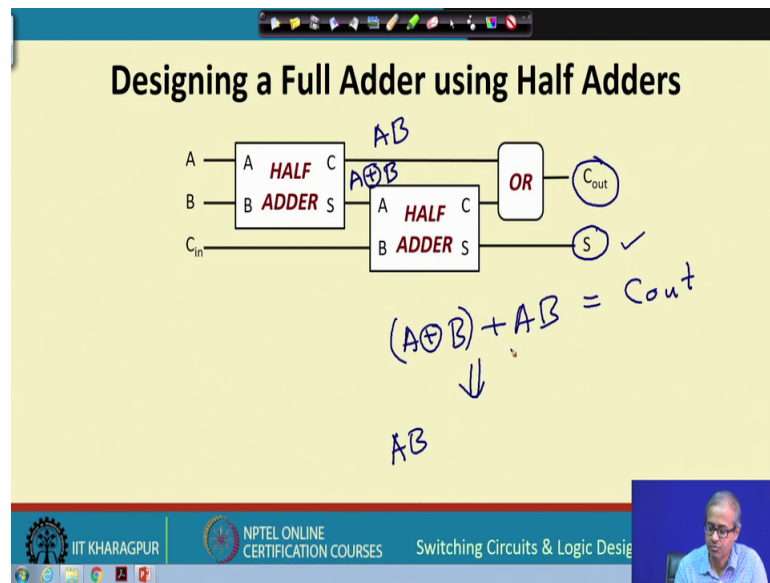
Let us check. So, what is the value here? It is XOR of A B ok. What is XOR of A B A bar B or A B bar? This is XOR of A and B. In this AND gate where ending this with C in ok. Just for convenience them writing only C, C in for simplicity and the second gate is just the AND of A B, so OR A B. This is what I get. Let us see what this expression is. The first one is A bar B C or A B bar C or A B.

Now, you recall we have A B bar C and A B. So, what does this mean? A B bar A bar B C plus A B. If you take B common here, it will be A B bar C plus A. Now, you recall a rule

which you discussed earlier. This  $A \bar{C}$  or  $A$  is nothing, but  $C$  or  $A$  which means  $B C$  or  $A b$ .

So, I can write this only  $B C$ . So, exactly in the same way this  $A \bar{B} C$  and  $A B$  if you combine, take  $A$  common it will be  $B$  or  $C$ . So, it will be nothing, but  $A C$  and this  $A B$  is already there. So, you see you get the same expression for carry out. This is a simpler way using one extra gate and also smaller gate here or gate is also smaller to input right.

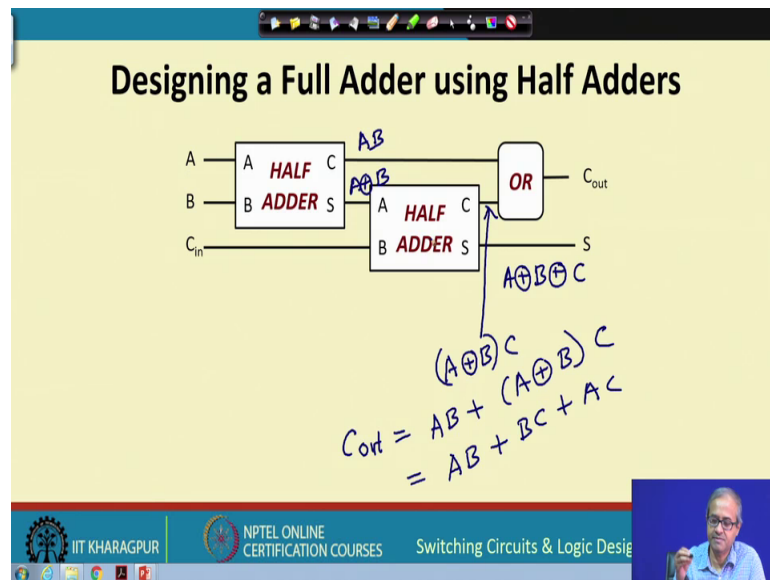
(Refer Slide Time: 22:17)



There is another way to implement a full adder that is using half adders. So, this I leave this as an excess for you to verify that whether this  $C_{out}$  and  $S$  that is in generated as correct or not because already you know what is the half adder. So, if  $A B$  are the inputs. The carry is nothing, but  $A B$  and sum is nothing, but  $A \text{ XOR } B$ .

In this half adder you are again applying the third input with  $A \text{ XOR } B$ . So, sum will be correct  $A \text{ XOR } B \text{ XOR } C$  sum is obviously, correct, but you have to check this  $A \text{ XOR } B \text{ OR } A B$ . This is what is carry out. So, this I will leave an excess for to verify whether this is actually equal to  $A B$ . This is the carry weight. This not  $A B$  actually no, no; I have just made a mistake. Let me correct it.

(Refer Slide Time: 23:33)



So, A B it is sum is A XOR B carry is A B. In the next half adder the inputs are A XOR B and C. Let us see.

So, the sum will be A XOR B XOR C and this carry output will be A XOR B and C. This point will be A XOR B. This is one input and this is second input. So, the carry out that we generate finally, this will be OR of these 2 week, OR of A B and this and I leave it as exercise for you to verify that this is actually equal to A B or B C or A C ok.

So, if you have a half adders as the building blocks, you can also design of full adder using that fine.

(Refer Slide Time: 24:37)

- Delay of a full adder:
  - Assume that the delay of all basic gates (AND, OR, NAND, NOR, NOT) is  $\delta$ .
  - Delay for Carry =  $2\delta$
  - Delay for Sum =  $3\delta$   
(AND-OR delay plus one inverter delay)

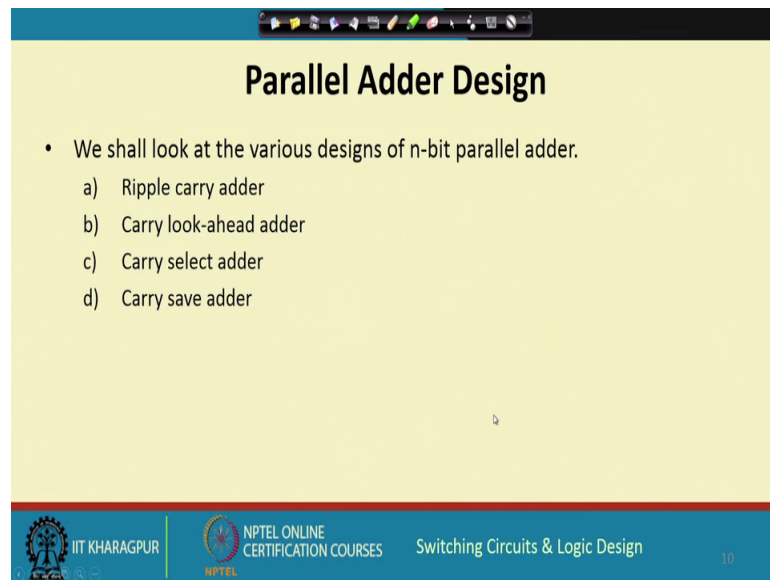
NOT AND OR

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES Switching Circuits & Logic Design

Now, using the previous methods; so, this delays. Already we have talked about earlier. So, here it is mentioned again. This is the original design of the full adder where this sum was generated by XOR function and the carry was generated using A B or A C or B C this way.

So, if delta is mentioned is delay the delay of a basic gate. So, far carry it requires 2 level of gates. First is AND, then OR. So, which is 2 delta and some XOR I said XOR can be implemented in 3 levels. First NOT then set of AND gates finally, OR gates OR gate 3. So, this will be 3 delta. So, delay of a full adder you can calculate like this. Carry will be generated after 2 delta, sum will be generated after 3 delta right.

(Refer Slide Time: 25:45)



The slide is titled "Parallel Adder Design" and contains a bulleted list of n-bit parallel adder designs. The footer includes the IIT Kharagpur logo, NPTEL Online Certification Courses logo, the course title "Switching Circuits & Logic Design", and the slide number "10".

- We shall look at the various designs of n-bit parallel adder.
  - a) Ripple carry adder
  - b) Carry look-ahead adder
  - c) Carry select adder
  - d) Carry save adder

So, now we shall see in our next lecture. So, what are the different ways to design parallel adders using these full adders and may be half adders also. So, we shall look at several different kinds of adders Ripple carry adder and Carry look-ahead adder in some detail and we shall briefly look at two other kinds of adders which are a little more complex. These are the little advanced kind of adders. We shall be looking at them.

So, this we shall be discussing starting from the next lecture onward. So, we come to the end of the present lecture where if you recall, we talked about the basic building blocks using which when our ready to design a parallel adder. We looked at the half adder. We will look at the half adder, the next step the full adder. Now, using this full adders and also half adders if you want, you can design a circuit that can add 2 in general n bit numbers. This is called a parallel adder.

This we shall be discussing in our next lecture.

Thank you.