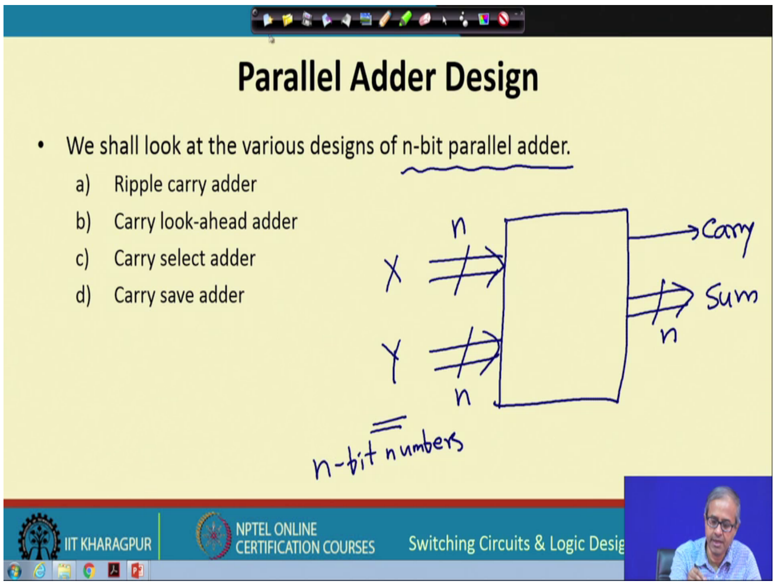
Switching Circuits and Logic Design

Prof. Indranil Sengupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 22
Design of Adders (Part – II)

So, in this lecture we shall be starting or discussion on how to design parallel adders. Earlier we had seen how we can design half adder and a full adder; we shall be using these as the basic building blocks to design this, so called parallel adders. So, the title of this lecture is Design of Adders the second part.

(Refer Slide Time: 00:38)



Parallel Adder Design

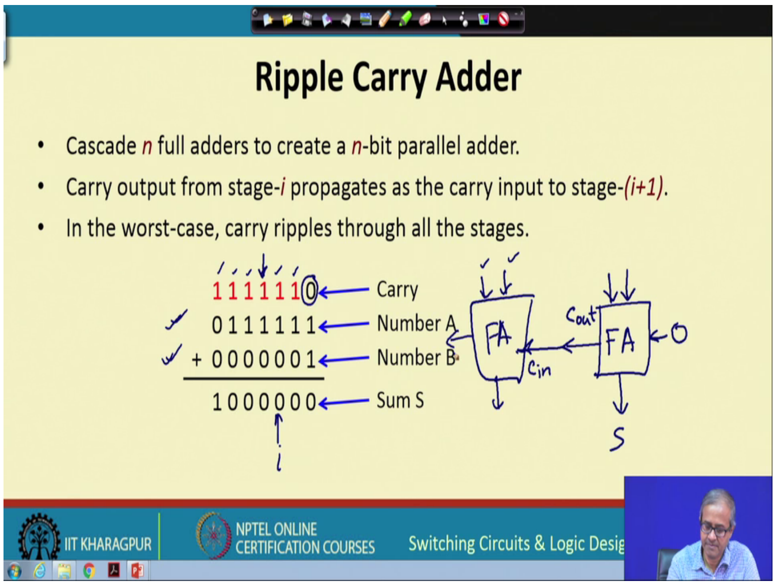
- We shall look at the various designs of n-bit parallel adder.
  a) Ripple carry adder
  b) Carry look-ahead adder
  c) Carry select adder
  d) Carry save adder

X
Y
n
n
n
Carry
Sum
n-bit numbers
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Desig

So, we mentioned in the previous lecture also that, we shall be looking at various designs of n-bit parallel adder. So, what is an n-bit parallel adder it means that I have a circuit, where there are 2 inputs I am applying let us call them a b x y whatever you call let us say one is X and one Y, both of these are n-bit numbers we show it like this. If you cut it and write n these means these X and Y are n-bit numbers.

This is shortcut way of showing it and in the output we generate sum, this will also be an n-bit and carry is a single bit. This is what you mean by an n-bit parallel adder we want to design a circuit like this fine.

(Refer Slide Time: 01:44)



Ripple Carry Adder

- Cascade *n* full adders to create a *n*-bit parallel adder.
- Carry output from stage-*i* propagates as the carry input to stage-*(i+1)*.
- In the worst-case, carry ripples through all the stages.

So, the first design that we look at is something called ripple carry adder and it very naturally and directly mimics our hand computation the (Refer Time: 01:57) add by hand. We at the bit stage by stage we accumulate the carry, the carry goes to the next stage the carry gets added with the 2 input bits are the next stage again carry is generated it goes to the next stage and so on ok.

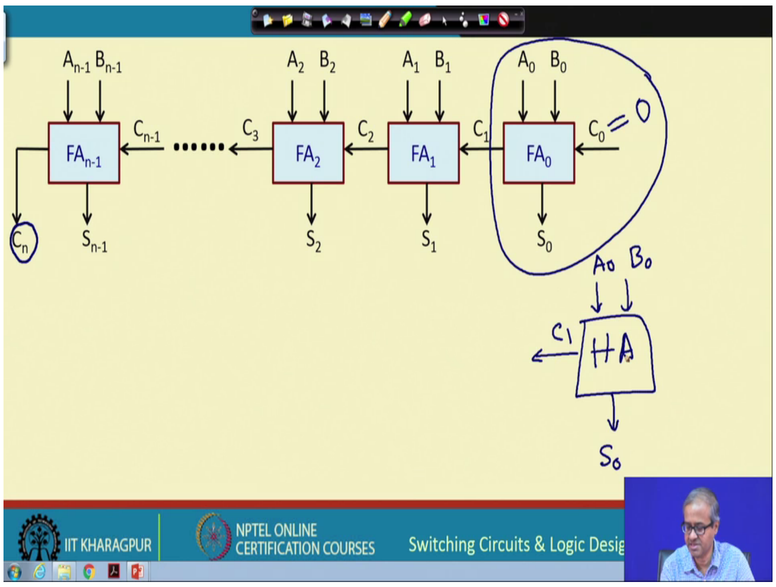
So, the ripple carry adder essentially just works on this principle we take n fuller; that means, we want to design an n-bit adder, we take n full adders and simply cascade them connect them 1 after the other. So, I shall show you how their connected and the connection should be such that the carry output from particular stage, let us say the i th stage must be applied as the carry input to the next stage.

This is how we do addition by hand. So, in the last lecture we took an example that that we can have an example where the carry can ripple through all this stages during addition. So, let us show that example once more. So, if you have two numbers like this that you are adding. So, initially this is no carry I am assuming 0. So, I am adding them up, the carry is going here, I am adding this 3 numbers again carry is going here I am adding them up.

So, the carry is going to the next stage. So, if I call any 1 of the stage as stage i. So, the carry of stage i that is generated will be going as the carry input of stage i plus 1 right. The sum is generated and the carry will be generate. So, you see if I have full adder now, if I have full adder like this, but the 2 inputs that are coming let us say the 2 input bits are applied here, I get the sum that sum bit is generated and whatever the carry is generated the carry this is my carry out and this is your carry in right. And if use another full adder site by site this carry out will be connecting as the input carry into the next stage.

So, your inputs are again coming the next sum you generating. So, exactly what you are doing by hand your trying to do like this. See for the first stage because the carry is 0 will be applying the 0 here right and for all other stages the carry in will be coming from the carry out of the previous full adder, and the 2 inputs will be applied here and will be getting the sum bits here, and the carry bits will be generated here right.

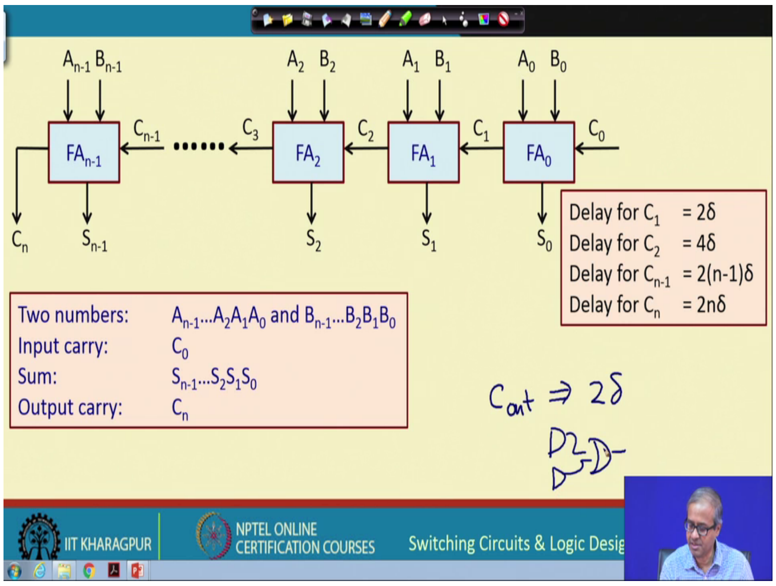
(Refer Slide Time: 05:05)



So, this is how an n-bit full adder looks like. You see there are n number of stages there are n full adders F A 0, F A 1 up to F A n minus 1, the 2 input numbers are A 0 to n minus 1 and B 0 to B n minus 1, Cc 0 is the input carry sum is generated S 0 to S n minus 1 and final carry outs C n is also generated.

Now, just the point to notice that well if this n-bit adder your using the standalone more, which means this carry input is always 0, then this first stage of this addition which is a full adder, you can replace it by a half adder; Because half adder if you recall has only 2 inputs. So, this will only have A 0 and B 0 A 0 and B 0 and it will be generating S 0 and it will be producing C 1 as the carry out ok. So, if we have an adder added design where there is no carry input, then you can replace the last stage by an half adder which will be a little less expensive because number of gates in half adder is less then that of full adder fine.

(Refer Slide Time: 06:35)



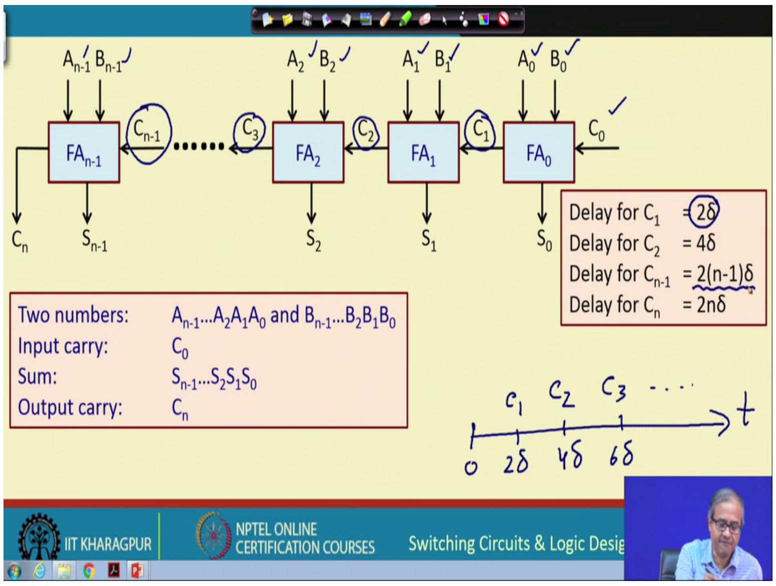
Delay for C_1	$= 2\delta$
Delay for C_2	$= 4\delta$
Delay for C_{n-1}	$= 2(n-1)\delta$
Delay for C_n	$= 2n\delta$

Two numbers:	$A_{n-1}\dots A_2A_1A_0$ and $B_{n-1}\dots B_2B_1B_0$
Input carry:	C_0
Sum:	$S_{n-1}\dots S_2S_1S_0$
Output carry:	C_n

So, what you have said is this there are two numbers were adding, one is A, another is B both are n-bit numbers input carry is C 0, sum is this and output carry is C n. Now let us look at the delay calculation, first look at the carries you recall for a full adder we have said the delay for a carry; that means, the carry out delay is twice delta.

Because there are some and gates followed by an OR gate there is two level circuit delta plus delta. That is why if you see here we have seen that suppose A 0 B 0 and C 0 has been applied at time T equal to 0; So, when will C 1 be generated the carry out of this after time 2 delta.

(Refer Slide Time: 07:38)



Delay for C_1	$= 2\delta$
Delay for C_2	$= 4\delta$
Delay for C_{n-1}	$= 2(n-1)\delta$
Delay for C_n	$= 2n\delta$

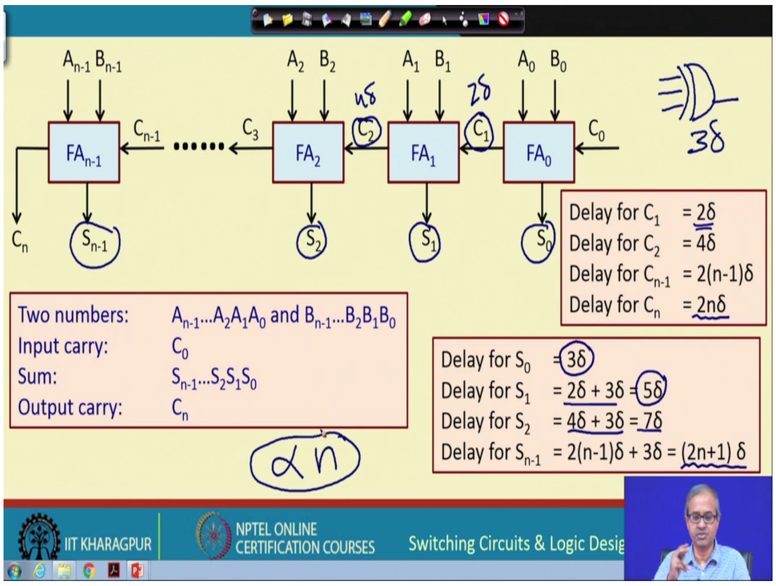
Two numbers:	$A_{n-1}...A_2A_1A_0$ and $B_{n-1}...B_2B_1B_0$
Input carry:	C_0
Sum:	$S_{n-1}...S_2S_1S_0$
Output carry:	C_n

So, if you look at the time suppose this is your time. So, this is time t equal to 0 a time t equal to 0 all the inputs have come all the inputs have been applied all the inputs and also C 0.

So, at a time twice delta, C 1 will be generated. So, at time twice delta this new value of C 1 will be fed as an input to F A 1. So, even though A 1 B 1 is applied earlier, but the carry input is coming now at 2 delta. So, this will be requiring another 2 delta time to calculate C 2. So, C 2 will be generated at 4 delta time C 2. Similarly C 3 will be generated at 6 delta time right.

So, in this way if you continue then C n minus 1 will be generated after twice n minus 1 delta and the final carry output 2222 like that it will be generated after twice n delta right. This is the total delay for the carry now let us see what will be total delay for this sum.

(Refer Slide Time: 08:55)



Two numbers:	$A_{n-1}...A_2A_1A_0$ and $B_{n-1}...B_2B_1B_0$
Input carry:	C_0
Sum:	$S_{n-1}...S_2S_1S_0$
Output carry:	C_n

Delay for C_1	$= 2\delta$
Delay for C_2	$= 4\delta$
Delay for C_{n-1}	$= 2(n-1)\delta$
Delay for C_n	$= 2n\delta$

Delay for S_0	$= 3\delta$
Delay for S_1	$= 2\delta + 3\delta = 5\delta$
Delay for S_2	$= 4\delta + 3\delta = 7\delta$
Delay for S_{n-1}	$= 2(n-1)\delta + 3\delta = (2n+1)\,\delta$

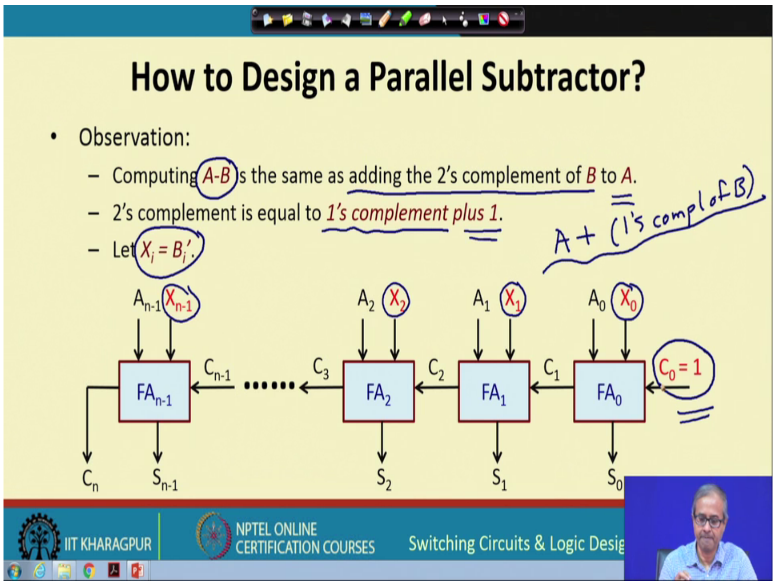
Now, you recall for full adder the delay of a sum is 3 delta because there is an XOR function and XOR is NOT, AND and OR. So, it requires twice delta. So, assuming that again all the inputs are coming time 0, this sum S 0 will be generated after time 3 delta because it is an XOR of a 0 B 0 and C C 0.

Now, what about S 1 you remembered just now what you said here that C 1 was generated at time twice delta. So, it will take another 3 delta time, now A 1 B 1 and C 1 is come at 2 delta it will take another 3 delta time. So, at 5 delta this s 1 will be available. Now again for the next stage, C 2 is generated at 4 delta. So, S 2 will be 4 delta plus another 3 delta for this addition.

So, 7 delta. So, it goes like this. So, finally, for S n minus 1 if you just see expand this, (Refer Time: 10:16) be twice n minus 1 delta plus 3 delta which is 2 n plus 1 delta. So, you see the delay of the carries two n delta delay of the sum is a little more 2 n delta plus another delta. So, one characteristic of this kind of ripple carry adder is that, the total delay is propositional to the number of bits you see 2 n delta 2 n plus 1 delta something like that.

So, what it means if I want to design very large adder see 8 bit adder, 16 bit, 32 bits, 64 bits then my addition time will go up. 16 bit adder will have double the delay as compare to an 8 bit adder. 32 bit adder will have 4 times delay as compare to an 8 bit adder. So, it is not so, easy to expand the number of bits in an addition without scarifying on the delay this is 1 main draw back here.

(Refer Slide Time: 11:24)



How to Design a Parallel Subtractor?

- Observation:
  - Computing A-B is the same as adding the 2's complement of B to A.
  - 2's complement is equal to 1's complement plus 1.
  - Let $X_i = B_i'$.

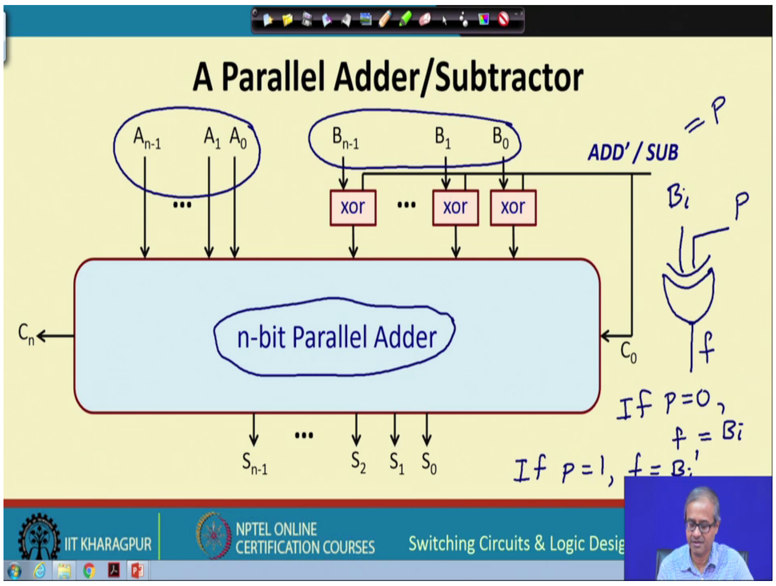
Now, let us see how we can design a subtractor now let now that you know that how we can design ripple carry adder. So, for parallel subtractor subtraction, we rely on or that old number theoretic knowledge number system we talked about, we talked about the 2's complement number system.

We said when you subtract two numbers it is nothing, but you take the 2's complement of the second number and add that is as good as subtracting ok. So, when you say that we are trying compute A minus B, we can do this same thing by adding the 2's complement of B to A and what is 2's complement? It is 1's complement plus 1 and what is 1's complement? You are doing not of all the bits let us say X i denote the naught of B i. Let us look at an arrangement like this. So, what we have shown here is nothing, but a ripple carry adder you see same ripple carry adder. But the only difference is this second input instead of applying B 0 B 1 B two we are applying X 0 X 1 X 2 and X n minus 1.

And what are these Xs? These are not of the corresponding B values which means, we are adding A with the 1's complement of B 1's complement of B right because 1's complement means just not, but we 1 to, but we have to add 2's complements right. So, what we do, we also add a 1 to the carry input. So, effectively what is happening? I am adding the 1's complement of B from the top and an additional 1 I am adding us carry input, which means it is becoming 2's complement. So, whatever this sum is coming out it is actually this same as A minus B. So, I am I mean able to carry out subtraction just by taking the naught of the second number and by feeding 1 in the carry input it is as simple as that right.

So, subtraction is very easy we do not need a separate subtract for that, we can use this same addition hardware same addition circuit with small adjustments ok. Now let us see whatever we have shown here how we can make it general. General means circuit, which can do both addition and also subtraction. So, let us see how that circuit will look like.

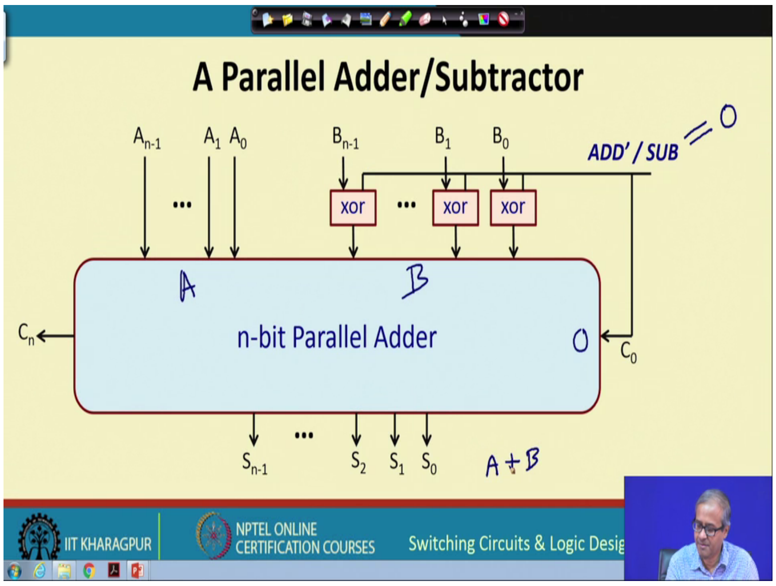
(Refer Slide Time: 14:33)



The circuit will look like something like this, where this is our adder this can be ripple carry adder let us say we have seen only one kind of adder so far, this can be any kind of an adder let us say ripple carry adder. So, what we have doing, the first input A we are applying to this adder, this second input actually what you are applying this is second input B ok, but you are not applying B directly there are some XOR gates.

So, there are some exclusive or gates what you doing, let us say the i th bit B i and the second input of this XOR gate, we are applying control input let us call it this add subtract let us in short call it P let us call it P. So, how does XOR gate behave I talked about it earlier. It says if P equal to 0 if the output is f if P equal to 0 the output is nothing, but B i; but if P equal to 1 then the output becomes the naught of B i ok. So, it is controlled inversion now let us see what happens.

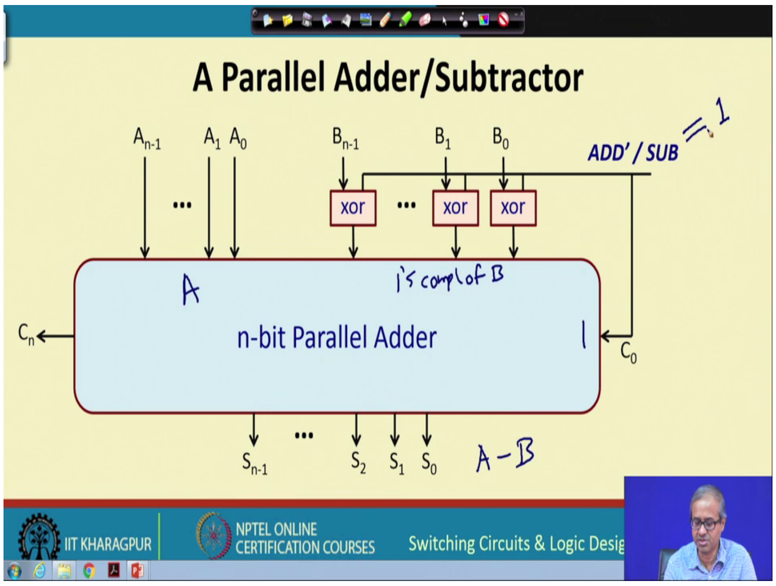
(Refer Slide Time: 16:14)



Now, let us suppose this add subtract control input I have applied 0; 0 means this XOR gates this Bs will pastrate.

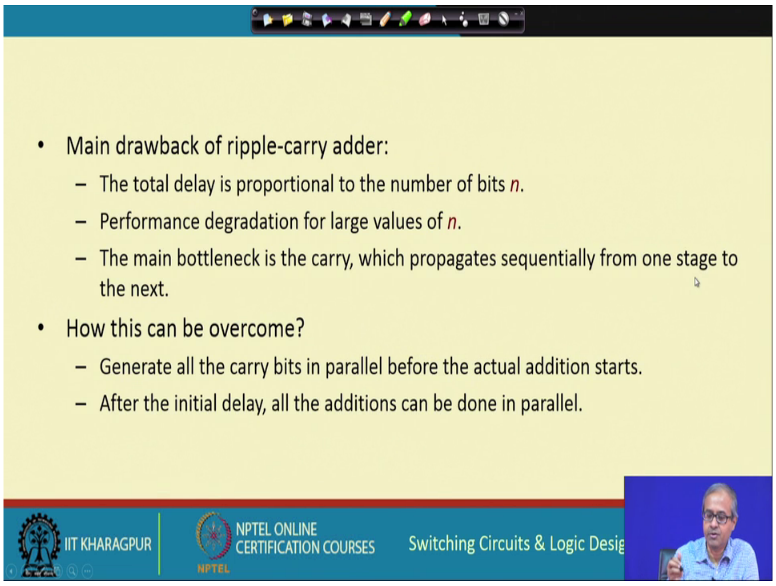
So, I have A here I have B here coming directly and carry input is also 0. So, this is just like addition normal addition and in the output will be getting A plus B right.

(Refer Slide Time: 16:44)



Now, suppose I apply add subtract control input 1. So, if I apply 1 what will happened? First thing is that XOR gates will be doing the naught of each of the B is. So, A will be applied here, but here it will be 1s complement of B 1s complement of B will come here and carry input is also 1. So, basically 2's complement will get added and what will get is A minus B. So, we have a circuit with the control input which if I set it to 0 it will be addition if I set it to 1 it will those subtraction ok. So, this is very general kind of thing just the addition subtraction circuit.

(Refer Slide Time: 17:32)



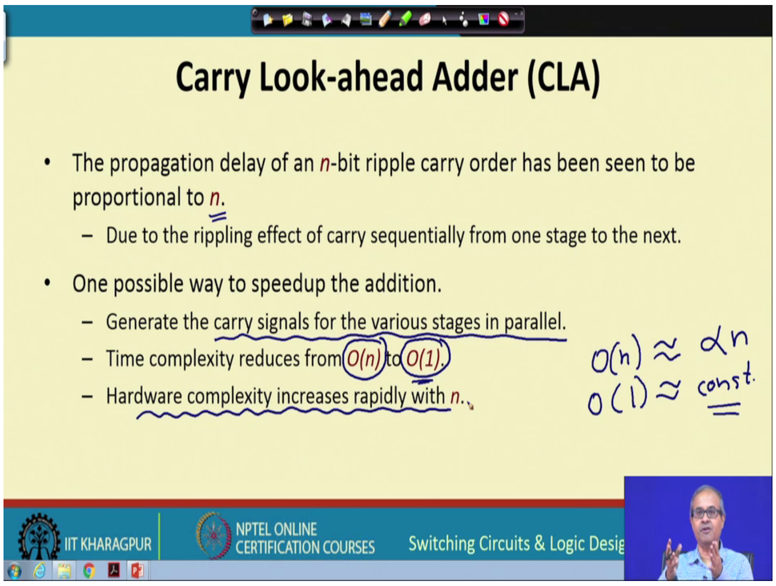
- Main drawback of ripple-carry adder:
  - The total delay is proportional to the number of bits *n*.
  - Performance degradation for large values of *n*.
  - The main bottleneck is the carry, which propagates sequentially from one stage to the next.
- How this can be overcome?
  - Generate all the carry bits in parallel before the actual addition starts.
  - After the initial delay, all the additions can be done in parallel.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL
Switching Circuits & Logic Desig

Now, let us look at what is the drawback of this ripple carry adder. Drawback of the ripple carry adder is that, the total delay we have seen earlier is propositional to n. So, as we increase the number of bits the delay also increases these the degradation in performance. But in reality we need to design adders that are pretty big 32 bits, 64 bit adding addition not only that we want to make them very fast we want to carry out the addition in a faster way.

So, how do you do this? Well one alternative maybe that if possible you generate all the carry bits and parallel, why because in a ripple carry adder the main bottle neck was the carry which was propagating sequentially from one stage to the next. So, somehow if you can generate all the carry bits in parallel before the actual addition starts, then after that all the additions can be done in parallel because we have generated all the carries and we did not have to wait any further for the carries, to propagate. This is the basic idea behind the next kind of adder that we propose that we shall discuss which is much more efficient in terms of the speed in terms of the delay.

(Refer Slide Time: 19:05)



Carry Look-ahead Adder (CLA)

- The propagation delay of an *n*-bit ripple carry order has been seen to be proportional to *n*.
  - Due to the rippling effect of carry sequentially from one stage to the next.
- One possible way to speedup the addition.
  - Generate the carry signals for the various stages in parallel.
  - Time complexity reduces from *O(n)* to *O(1)*.
  - Hardware complexity increases rapidly with *n*.

$$O(n) \approx \alpha n$$
$$O(1) \approx \underline{\underline{\text{const.}}}$$

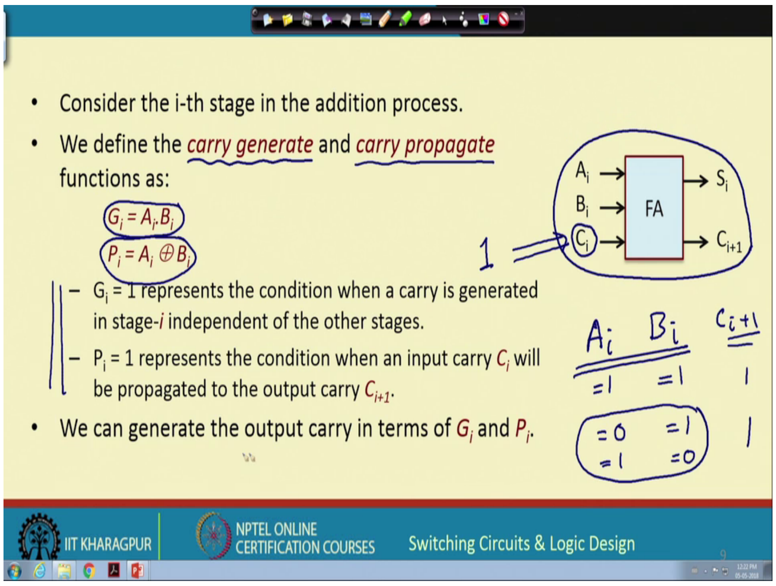
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Desig

This is called carry look ahead adder. Let us see the basic idea what carry look ahead adder is we have mention this in the previous slide also

That the propagation delay of ripple carry adder is proportional to n, but in a carry look ahead adder we are generating the carry signals in parallel. This is notation which we use to denote complexity, but you need not have to worry about the exact meaning of this, this order n roughly indicates it is proportional to n and order 1 means it is constant.

So, in a ripple carry adder the total time taken is proportional to n. So, if you make bigger adder the time increases, but in a carry look adder what you are saying that the addition time is constant irrespective of the value of n. But the drawback is that hardware complexity number of gates, size of gates, they will increase very rapidly with the value of n. This actual limits the value of n to certain value beyond that a carry look ahead adder can become very complex.

(Refer Slide Time: 20:38)



- Consider the i-th stage in the addition process.
- We define the *carry generate* and *carry propagate* functions as:
  - $G_i = A_i.B_i$
  - $P_i = A_i \oplus B_i$
  - $G_i = 1$ represents the condition when a carry is generated in stage-*i* independent of the other stages.
  - $P_i = 1$ represents the condition when an input carry C_i will be propagated to the output carry C_{i+1}.
- We can generate the output carry in terms of G_i and P_i.

Let us see how it works. So, we look at the full adder again look at the ith stage of the full adder. Just with respect to the ripple carry adder, you just imagine that in the ith stage the inputs were address A i B i and the carry inputs C i and the outputs were sum S i and the carry C i plus 1, which was going to the next stage. Now we define two functions call carry generate and carry propagate.

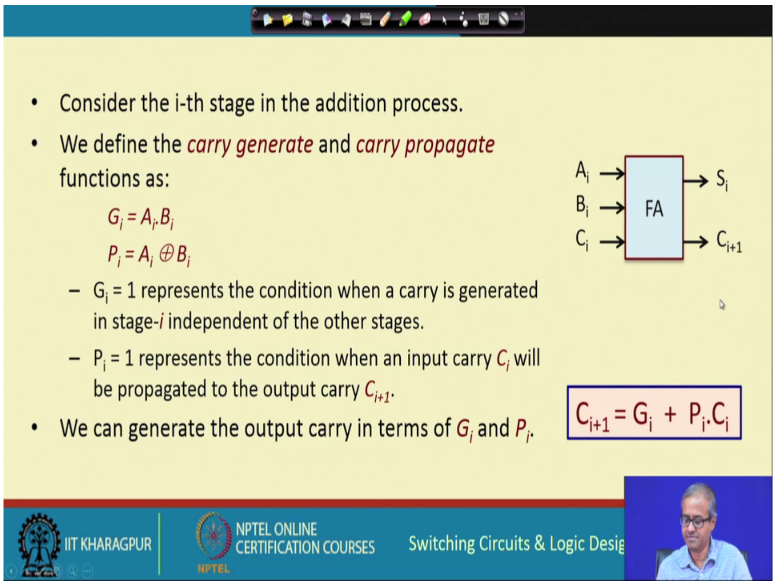
Will in terms of the inputs A i and B i only these are may 2 input bits which I am trying to add in the ith stage, first time calling carry generate. Generate means under what conditions of A i and B i carry will always be generated. You see whenever A i is equal to 1 and B i is equal to 1 then irrespective of what my carry input is this carry out will always be equal to 1 this C i plus 1 will always be equal to 1 because if this is 1 1, if carry input is 0 then also there will be a carry. If carry 1 then also they will be carry.

So, I can express this as the AND function and of A and B if it is 1 1 then only it is 1 and the second one is carry propagate. Carry propagate says that suppose there is an input carry C i is 1. So, under what condition this 1 will propagate to the output; that means, there will also be an output carry; propagate means under what conditions.

So, A i and B i this 1 will propagate. You see if it is a 1 1 then it is generating carry anyway. But for propagation the two condition will be one of them will be 1 either 0 1 or 1 0. If it is 1 0 1 then also there will be a carry out if it is 1 1 0 then also they will be carry out. So, this is the condition for carry propagation 0 1 and 1 0 is nothing, but the exclusive are function carry out will be 1 if A 0 B 1 or A 1 B 0. So, you generate is your and propagate is xor, this is this is what I have just now explained.

Now, the output carry C i plus 1, we can now generate or express in terms of P i and G i. How?

(Refer Slide Time: 23:35)



- Consider the i-th stage in the addition process.
- We define the *carry generate* and *carry propagate* functions as:

  $G_i = A_i.B_i$

  $P_i = A_i \oplus B_i$

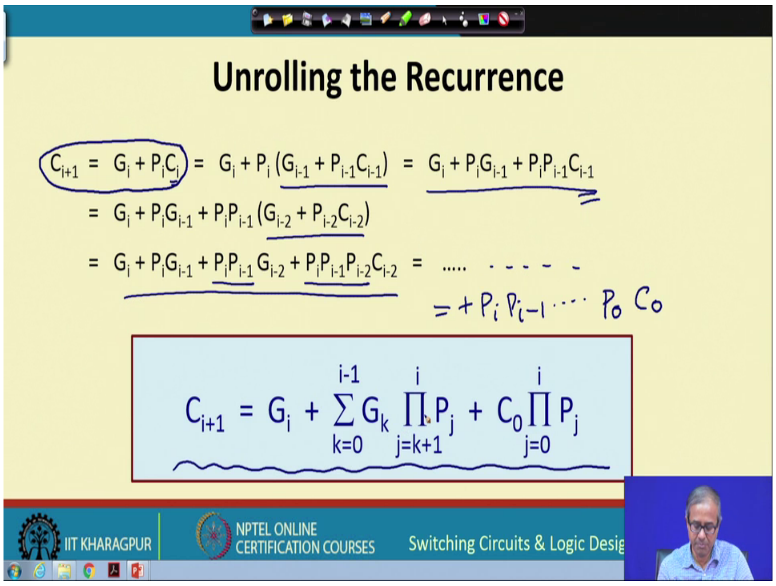
  - $G_i = 1$ represents the condition when a carry is generated in stage-*i* independent of the other stages.
  - $P_i = 1$ represents the condition when an input carry C_i will be propagated to the output carry C_{i+1}.
- We can generate the output carry in terms of G_i and P_i.

$$\boxed{C_{i+1} = G_i + P_i.C_i}$$

A_i
B_i
C_i
FA
S_i
C_{i+1}
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Desig

We can simply write the carry output is either there is carry generated or the carry propagation condition is true and C i is 1 there is a input carry. So, only under this condition C i plus 1 will be 1 means either carry is generated this A i B i both are 1 or P i is 1 means I mean exactly one of them are 1 and C i is also 1 right.

(Refer Slide Time: 24:08)



Unrolling the Recurrence

$$C_{i+1} = G_i + P_iC_i = G_i + P_i(G_{i-1} + P_{i-1}C_{i-1}) = G_i + P_iG_{i-1} + P_iP_{i-1}C_{i-1}$$
$$= G_i + P_iG_{i-1} + P_iP_{i-1}(G_{i-2} + P_{i-2}C_{i-2})$$
$$= G_i + P_iG_{i-1} + P_iP_{i-1}G_{i-2} + P_iP_{i-1}P_{i-2}C_{i-2} = \ldots\ldots$$
$$= + P_i P_{i-1} \cdots P_0 C_0$$

$$\boxed{C_{i+1} = G_i + \sum_{k=0}^{i-1} G_k \prod_{j=k+1}^{i} P_j + C_0 \prod_{j=0}^{i} P_j}$$

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Desig

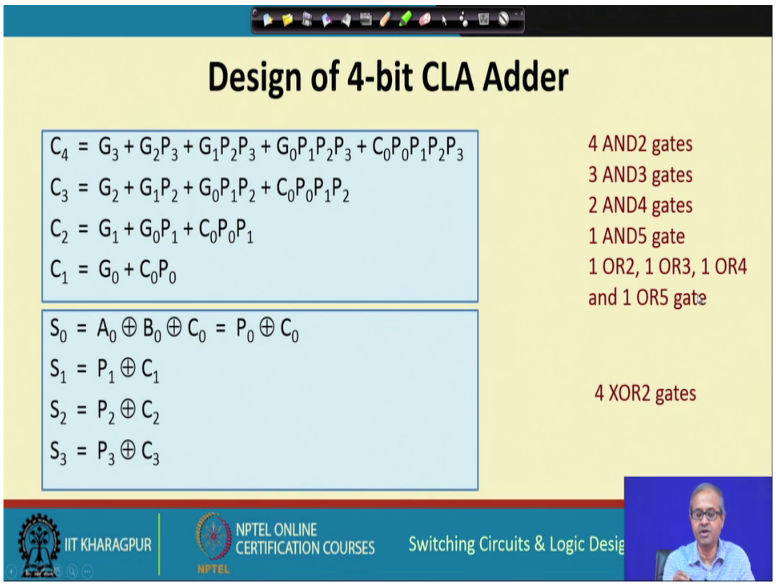
So, we have this relation now if you go and expanding this C i plus 1 equal to this. The C i I can write like this multiply this out I get this, this C i minus 1 I can again write like this I again multiply go and doing this to this you can go on continuing.

So, if you go on continuing. So, you go for I i minus 1 i minus 2 so on you go up to the last stage. So, the last term in the last term what will happened? This P i P minus 1 it will go on P i P i minus 1 so, on the last one will be P 0 and then you will be having C 0 this will be the last plus something.

So, this whatever your getting you can check this you can express this in a concise form like this C i plus 1 equal to G i will be there last term will be there, last term is this as I said C 0 multiply with by the product of all the P 0's Pi is then the intermediate terms, there will be summation of some G k and then the products like you are said P 1 P to P i minus 1 P and P 2 like this.

So, it will be something like this, but you really do not need to understand or remembered this in that way.

(Refer Slide Time: 25:48)



Design of 4-bit CLA Adder

$C_4 = G_3 + G_2P_3 + G_1P_2P_3 + G_0P_1P_2P_3 + C_0P_0P_1P_2P_3$

$C_3 = G_2 + G_1P_2 + G_0P_1P_2 + C_0P_0P_1P_2$

$C_2 = G_1 + G_0P_1 + C_0P_0P_1$

$C_1 = G_0 + C_0P_0$

4 AND2 gates
3 AND3 gates
2 AND4 gates
1 AND5 gate
1 OR2, 1 OR3, 1 OR4 and 1 OR5 gate

$S_0 = A_0 \oplus B_0 \oplus C_0 = P_0 \oplus C_0$

$S_1 = P_1 \oplus C_1$

$S_2 = P_2 \oplus C_2$

$S_3 = P_3 \oplus C_3$

4 XOR2 gates

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

So, what we want is that in terms of our design, suppose where trying to design a 4 bit carry look ahead adder. So, just using this expressions that we have just now talked about if we expand them like this same way. So, I am looking from the reverse way, the last stage C 1 can be written as G 0 plus C 0, P 0 C 0 is already there you already know the carry input. C 0 is the carry input of the last stage right C 0 and all the A B inputs are given to you.

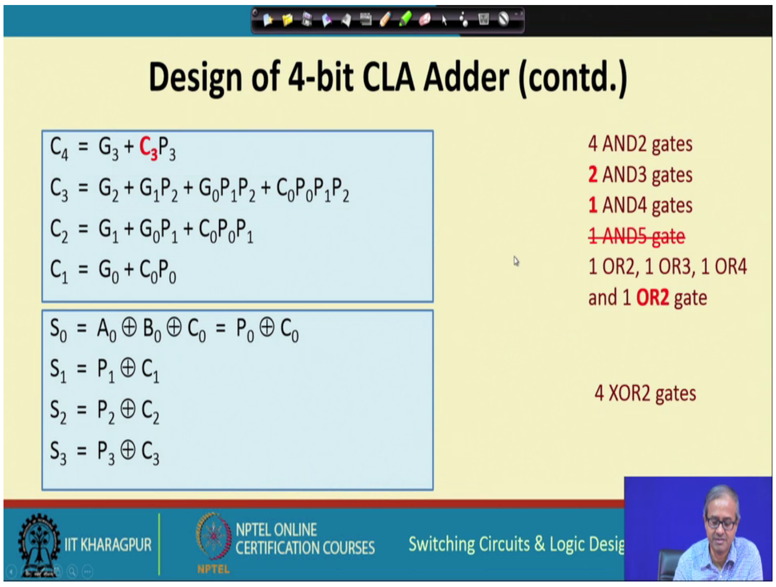
So, C 1 you can generate directly using this equation. Now C 2 if you just apply that previous one that expansion C 2 will be like this, this you can verify C 3 will be like this and C 4 will be like this. And this sums sum as we know sum is nothing, but the XOR of the 3 inputs in a full adder.

Now, A 0 XOR B 0 is nothing, but the carry propagate function. So, propagate XOR of the carry is nothing, but the sum. So, you can generate this sums like this. So, you see what you need. So, in order to do this unit 4 AND2 means 2 input and AND2 refers to 2 input AND gates.

You see there is one here or here 2 input AND gates, 3 3 input AND gates one is here one is here one is here, 2 4 input AND gates one is here one is here and 1 5 input AND gate it is here. And similarly for to do this OR operations, you need 1 2 input OR for C 1 1 3 input OR for C 2, 1 4 input OR for C 3 and 1 5 input OR for C 4 and for sum you need 4 XOR 2 input XOR’s right.

Now, it is possible to simplify this expression little bit if you make an observation the observation is us follows, you see C 3 is this ok. Now, in C 4 if you take consider this part and if you take P 3 common, you see whatever remains G 2 plus G 1 P 2 G 0 P 1 P 2 C 0 P 0 P 1 P 2 is nothing, but this. So, you can write this as P 3 multiplied by C 3. So, your expression will become much simpler.

(Refer Slide Time: 28:42)



Design of 4-bit CLA Adder (contd.)

$C_4 = G_3 + C_3P_3$

$C_3 = G_2 + G_1P_2 + G_0P_1P_2 + C_0P_0P_1P_2$

$C_2 = G_1 + G_0P_1 + C_0P_0P_1$

$C_1 = G_0 + C_0P_0$

4 AND2 gates
2 AND3 gates
1 AND4 gates
~~1 AND5 gate~~
1 OR2, 1 OR3, 1 OR4 and 1 OR2 gate

$S_0 = A_0 \oplus B_0 \oplus C_0 = P_0 \oplus C_0$

$S_1 = P_1 \oplus C_1$

$S_2 = P_2 \oplus C_2$

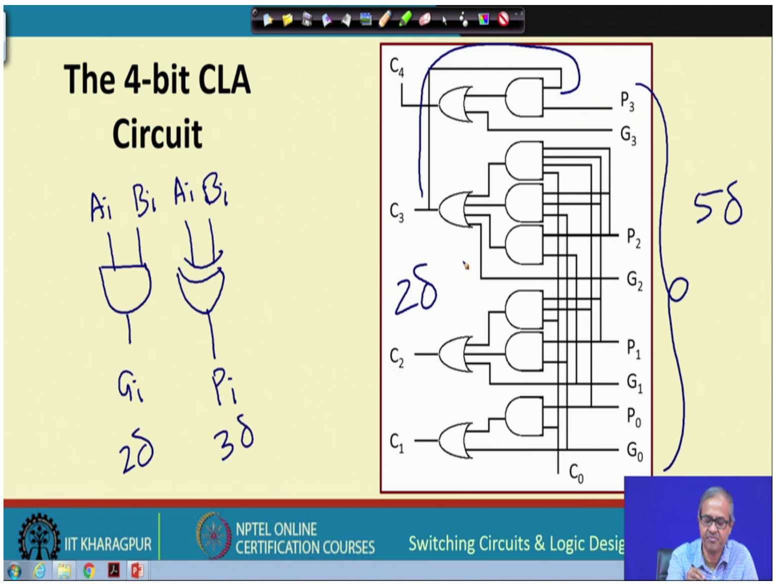
$S_3 = P_3 \oplus C_3$

4 XOR2 gates

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

You can write it like this straight away it becomes simpler and accordingly the size of the gates also becomes smaller this 5 input and gate is no longer required at all. So, you are cost reduces right.

(Refer Slide Time: 29:04)


The 4-bit CLA Circuit
A_i B_i A_i B_i
G_i P_i
2δ 3δ
C_4 P_3 G_3 C_3 P_2 G_2 C_2 P_1 G_1 C_1 P_0 G_0 C_0
2δ
5δ
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Desig

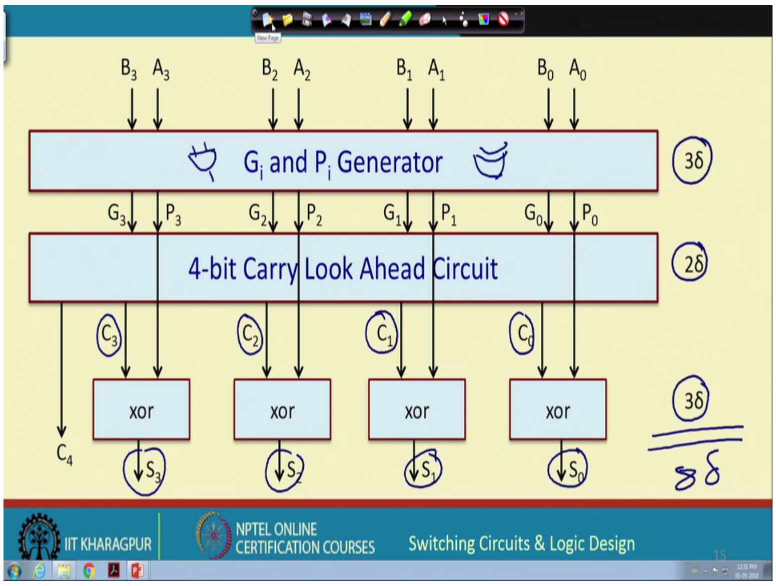
So, here for this 4 bit carry look ahead adder, this expression whatever means we have done here generated if you show the equivalent gate level realization it is sum what like this.

So, I am assuming that already P i G i are already generated because P i if you recall G i is nothing, but an and gate if you have A i and B i the two inputs, this will be G i and the propagate function is XOR gate A i B i this is P i. So, I am assuming this P 1 P 2 P 3 all these propagated and generate has already been generated by this, this can be done in parallel all the stages can generate the P i and G i in parallel this have been generated, then for the carry generation I need this circuit.

Well this is after minimization C 3 is being fed here right, but if you do not do minimization it will be two levels only. So, without minimization the delay will be little more because 2 again plus 2 4. But without minimization, this circuit requires delay of twice delta and followed by or and for P i G i for G i it is twice delta for P i XOR it is thrice delta.

So, this is more. So, the total delay will be 5 delta 2 delta to generate these 3 delta we generate this and 2 delta for this circuit.

(Refer Slide Time: 30:49)



B_3 A_3
B_2 A_2
B_1 A_1
B_0 A_0
G_i and P_i Generator
3δ
G_3 P_3 G_2 P_2 G_1 P_1 G_0 P_0
4-bit Carry Look Ahead Circuit
2δ
C_3 C_2 C_1 C_0
xor
xor
xor
xor
3δ
C_4
S_3 S_2 S_1 S_0
8δ
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Design

So, the overall picture looks like this, this is the this is 4 bit carry look ahead adder scheme. So, you see the 4 bits of the numbers the two numbers A and B or fed to the G i and P i generator. So, as I said G i is nothing, but an AND gate they will be 4 such AND gates and P i is nothing, but an XOR gate there will be 4 such XOR gates and because XOR gate has higher delay the delay of this stage will be 3 delta.

Then after you do this, just in the previous slide the diagram I showed the 4 bit carry look at circuit will be generating the carries in parallel and because is a two level and or circuit the delay is only two delta. Now, for generating the sum we do not need any full adder anymore because as we see saw from the expression this SS are nothing, but this propagate function XOR the carries. So, if you take P 0 XOR C 0, P 1 XOR C 1, P 2 XOR C 2 like this you can directly generate this sum and for XOR unit another 3 delta.

So, you see the total delay is 8 delta which is independent of the number of bits right. But the only point to notice that this you can see from the expressions that have generated earlier just let us go back this expressions. So, as you go on increasing number of bits the complexity of the expression will also go on increasing, the size of the gates number of gates will increase very rapidly this is the main drawback.

So, total time as I said will be 8 delta. So, with this we come to the end of this lecture, where we have talked about two different kinds of parallel adders ripple carry adder and also carry look ahead adder. So, we shall be continuing with our discussion in the next lecture.

Thank you.