Architectural Design of Digital Integrated Circuits

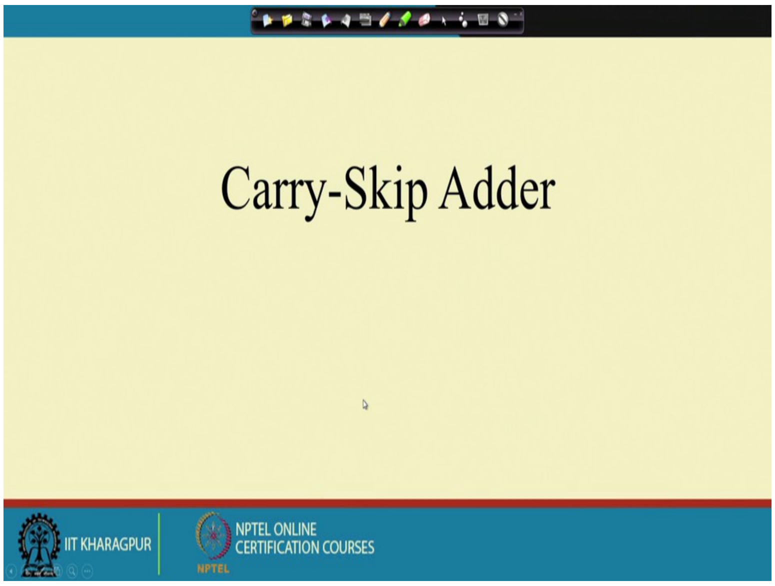
Prof. Indranil Hatai
School of VLSI Technology
Indian Institute of Engineering Science and Technology, Shibpur, Howrah

Lecture – 17
Efficient Adder Architecture (Contd.)

Welcome back to the course on Architectural Design of IC's. So, in the last class, we have seen that means, the carry ripple adder how we can design this ripple carry adder. Then, from that particular point how we can generate or that means, what is the logic behind generating the carry and then propagating the carry; that means, the main problem with ripple carry adder is that unless and until the carry is available on the that means, subsequent stages, it cannot start its computation.

So, that is the main problem and that is why we are getting a longer critical path if considering the bit that means, more of the bit for addition operation. So, that is why what we have seen? We have we have studied the operation table and from there that operation table, we have seen that the carry and gen, the carry is basically propagated and the carry is being generated based on the corresponding input bits, which is A and B and how they are related that we have seen in the last class.

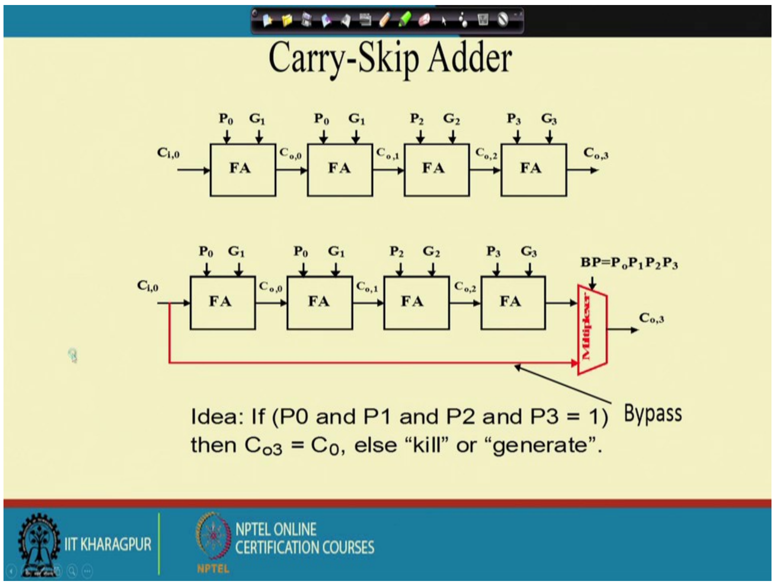
(Refer Slide Time: 01:27)



Carry-Skip Adder

So, today again we will see another that means, architecture of which is Carry-Skip Adder ok. So, Carry-Skip Adder the by its name it is says that that means, we can skip the carry ok. So, where we can skip the carry or how we can skip the carry or what is the that means, intention or what is the that means, usefulness of skipping the carry whenever we are performing the addition operation that we will see in this particular lecture.

(Refer Slide Time: 02:01)



Carry-Skip Adder

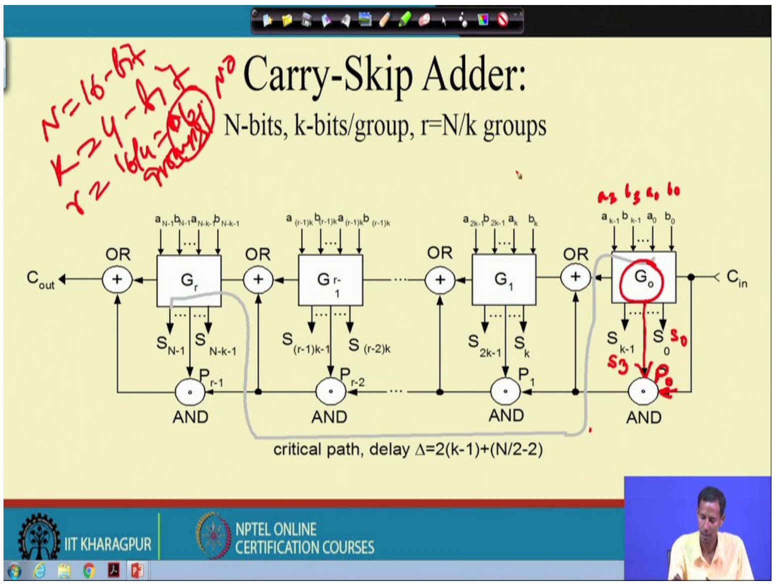
Idea: If (P0 and P1 and P2 and P3 = 1) then $C_{o3} = C_0$, else "kill" or "generate".

So, here you see the first the upper architecture is nothing but the ripple carry adder; but here if you just see there is I have put one multiplexer over here and that this input Ci, that means the whatever carry is coming over here that is basically connected at this particular point of this multiplexer. So, that means and what is the select line of this? The select line of this is this P 0 P 1 P 2 ANDed operation of P 0 P 1 P 2 and P 3. So, means what? So, what is P 0 and G; this P 0 and G?

This P and G that is the logic for propagation of the carry and G is the generation of the carry logic ok. So, now, if I just put this particular that means, if I just that means, this full adder cell if I just generate using the previous architecture; then at the final level or at this particular stage I put 1 2 is to multiplexer where if this all this propagated carry is 1; so at the time it will select the carry input of this Ci 0 as the output of C 3. So, here the idea is that if P 0 and P 1 and P 2 P 3 all are 1; so at that time C3 there will be the C 0 else kill or generate the corresponding carry.

So, if you just if I just want to design 1 in bit carry skip adder; so, at that time if I just make 1 group of k bits.

(Refer Slide Time: 04:02)



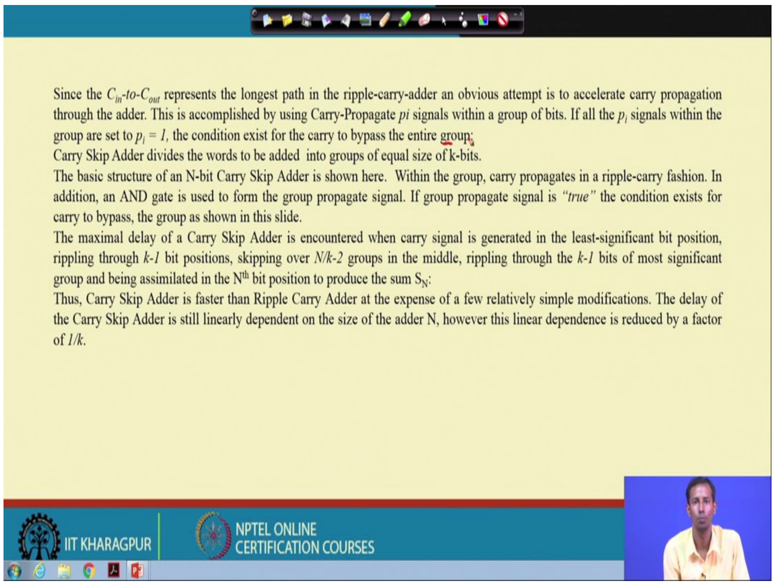
So, at that time I need N divided by k number of groups ok. So, N divided by k number of group. So, this groups if I just write it like G 0 G 1 G G r minus 1 and G r; where, r is the number of groups and each of this group is basically the Carry Skip Adder which is basically considering of k number of bits ok. So, then this carry that means, the corresponding carry out that is basically OR with this in each of these stages and it is being that means, the sum is generated from each of this group; where, this is a 0 to a k minus 1 ok.

That means, if I just consider if I want to design suppose a N if I consider of sixteen bit and k if that is of 4 bit ok; so that means, at that time I will have r equals to 16 by 4 that is equals to sorry 4 number of groups ok. So, each of this then this will be a 0 b 0 to a 3 b 3 ok. So, that means here it will be a 3 b 3 this will be upto a 0 and b 0.

So, here up to that means, how much I will get that is S 0 to S 3 and this is line is basically the corresponding that P 0 value with that means, this is the corresponding P 0 value which I am getting over here and this is the G 0; that means, this 4 of this I will get and this is basically OR with in each of these group bit is basically then or ok.

Now, if I just go to the; that means.

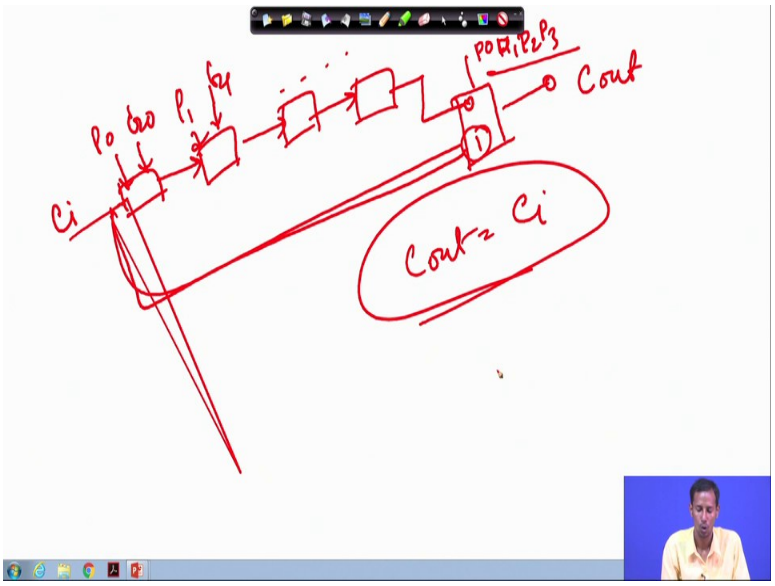
(Refer Slide Time: 07:01)



Since the C_{in}-*to*-C_{out} represents the longest path in the ripple-carry-adder an obvious attempt is to accelerate carry propagation through the adder. This is accomplished by using Carry-Propagate *pi* signals within a group of bits. If all the p_i signals within the group are set to $p_i = 1$, the condition exist for the carry to bypass the entire group;
Carry Skip Adder divides the words to be added into groups of equal size of k-bits.
The basic structure of an N-bit Carry Skip Adder is shown here. Within the group, carry propagates in a ripple-carry fashion. In addition, an AND gate is used to form the group propagate signal. If group propagate signal is *"true"* the condition exists for carry to bypass, the group as shown in this slide.
The maximal delay of a Carry Skip Adder is encountered when carry signal is generated in the least-significant bit position, rippling through *k-1* bit positions, skipping over *N/k-2* groups in the middle, rippling through the *k-1* bits of most significant group and being assimilated in the N^{th} bit position to produce the sum S_N:
Thus, Carry Skip Adder is faster than Ripple Carry Adder at the expense of a few relatively simple modifications. The delay of the Carry Skip Adder is still linearly dependent on the size of the adder N, however this linear dependence is reduced by a factor of *1/k*.

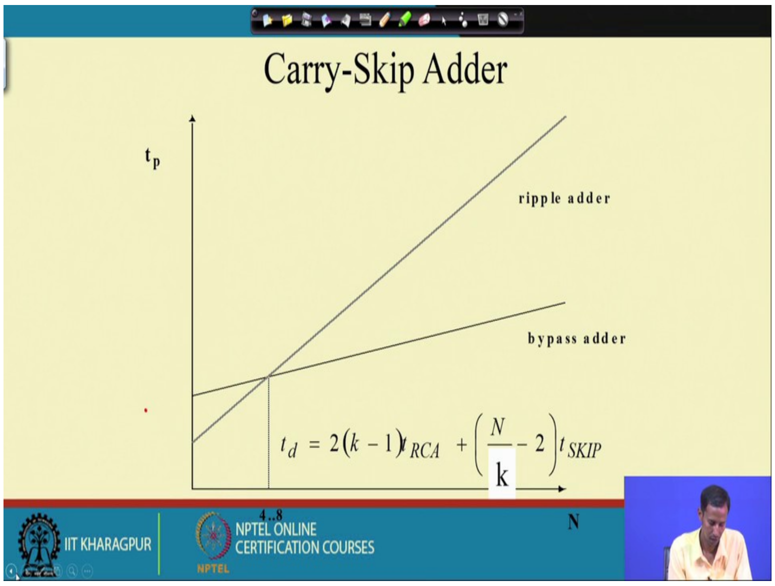
Then, if you just consider since the C in-to-C out represent the longest path in the ripple-carry-adder an obvious attempt is to accelerate carry propagation through the adder ok. So, this is accomplished by Carry-Propagate pi signals within a group of bits. If all the pi signals within the groups are set ai to pi equals to 1; then the condition exist for the carry to bypass the entire group.

(Refer Slide Time: 07:39)



So, in the last that means, in the last class what we have seen is that. Suppose, I am having this ok. So, this is my suppose P 0 and G 0; then this is P 1 and G 1 something like this which are connected, then if this is the C i, then that will be ultimately this is the P 0 P 1 P 2 and P 3 signal with which is basically con selecting this. If all the bits are 1; so at the time it is selecting this C o final out C out equals to this C i ok.

(Refer Slide Time: 08:39)



Carry-Skip Adder

$$t_d = 2(k-1)t_{RCA} + \left(\frac{N}{k} - 2\right)t_{SKIP}$$

So, considering this, considering this C into C out, considering this C into C out; now if I this each of this is basically nothing but that carry this 4 bit Carry Skip Adder if I consider a k equals to of 4 bit right and then this we have divided it like this.

So, the final C out is basically now we have find out something like this ok. So, now, the critical path will became or the critical path now it will traverse from this particular point to this and the total delay will be 2 k minus 1 plus N by 2 minus 2 that is the where k is the number of that means, bits per group and N is the number of bits what we have considered.

So, Carry Skip Adder divides the words to be adder into groups of equal size of k bits. The basic structure of N bit carry skip adder, we have already shown and within the group carry propagates in a ripple-carry fashion within the groups ok. But in addition, an AND gate is used to form the group propagate signal. If group propagate signal is “true”, they are condition exist for the carry to bypass, the group as shown in the corresponding slide.

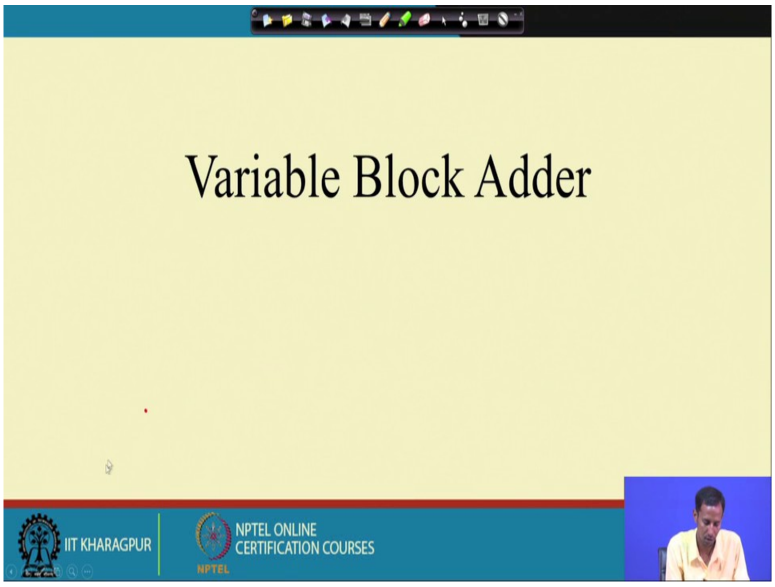
The maximal delay of the Carry Skip Adder is encountered when carry signal is generated carry signal is generated in the LSB position, rippling through k minus 1 bit position, skipping over N divided by k minus 2 groups in middle and rippling through k minus 1 bits of MSB group and being assimilated in the N the bit position to produce the sum of S N.

Thus, Carry Skip Adder is the faster than Ripple Carry Adder at the expense of a few relatively simple modification or it relatively addition of few extra gates the multiplexer or extra hardware. If I say the delay of the Carry Skip Adder is still linearly dependent on the size of the adder of N, however this linear dependence is reduced by a factor of k ok.

So, why this factor of k it is reduced because we have already divided by N divided by k and here you see as this depends on k and here the it is not that much linearly that means, in ripple carry adder how it looks that means, increases that is the delay will be like N minus 1 number of stages I need to cover; but here I do not need to cover that N minus 1 number of.

So, here the delay will be N by 2 minus 2 plus 2 into k minus 1 ok. So, the delay has been reduced or the critical path has been reduced by a factor of 4; if I consider k equals to 4 ok. So then, if you just see the curve; so at the time this ripple carry adder this is the that means, the corresponding values of N ok. So, this and this is the corresponding critical path ok. So, if I just plot whenever we are considering this we are increasing the number of stages.

(Refer Slide Time: 12:26)



Variable Block Adder

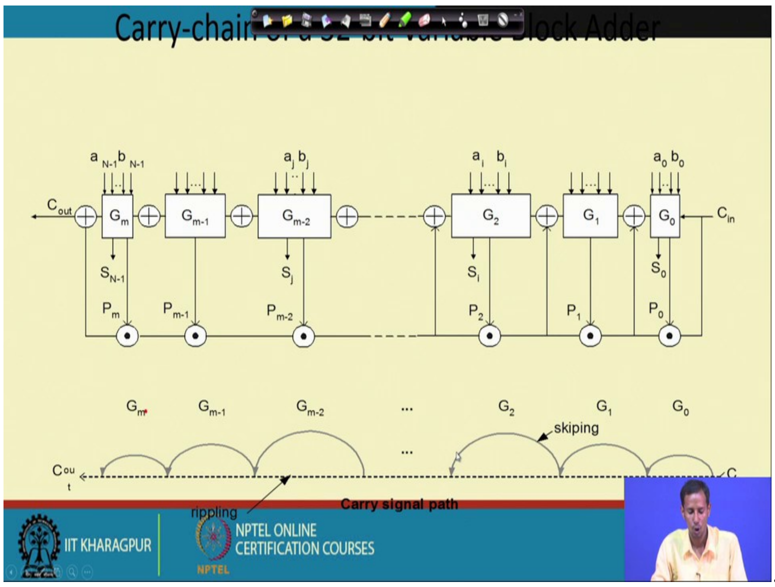
So, the critical path also increases linearly along with this; but for bypass adder, for bypass adder what happens? Initially, the delay is more ok. If you consider lower number of that means; but for more number of N as this is as increases; so at that time it is started bypassing and then, it is basically the delay is reduced for more if you consider more number of bits. So that means, what I can say that when this what I said in the previous slide is that this Carry Skip Adder is linearly dependent on the size of the adder N; however, this linear dependence is reduced by a factor of k.

So that means, for when the smaller values of N at that time, I will not get that much of benefit when you feed in carry skip adder; but whenever we will use or whenever I need to design the N is as large as possible. So, at the time as I can reduce the delay by a factor of k or so that means the linear dependency, at the time it got reduced and I can get the that means, the corresponding delay something like this. So, that is why you can use Carry Skip Adder for larger number of addition operation, where it requires ok. So, then next what we will see? We will see another; that means addition operation which is Variable Block Adder operation.

So, Variable Block Adder operation means so here, we are considering what? We are considering the k equals to that is fixed whether that is 4 or 3 or 2 or 6 something like this. But, if I just vary the bit width that means, if I vary the k at the time, what will be the effect of that ok? So, in the in the in the previous slide, if you just see that here you see the maximal delay of Carry Skip Adder is encountered when carry signal is generated in the LSB bit position ok.

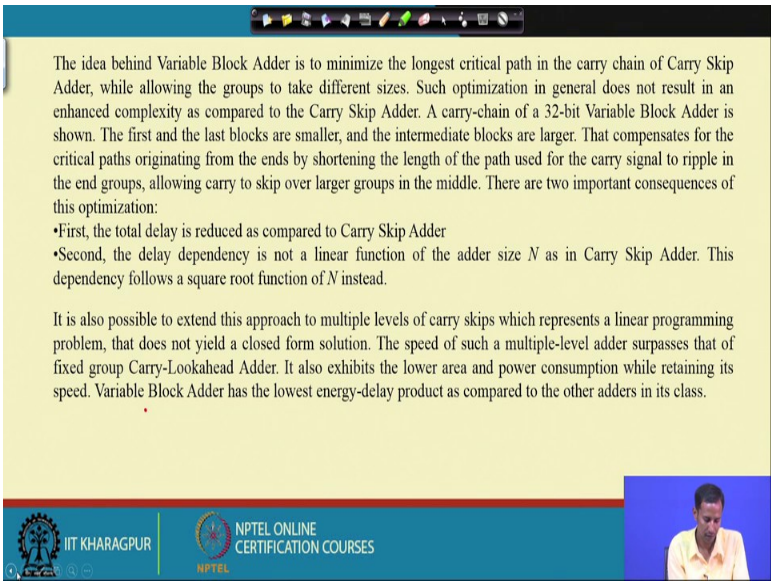
So, in the LSB bit position, if it in that means, generates the carry; so at the time it has to ripple through k minus 1 bit position and skipping over N minus k by 2 groups in the middle, rippling through k minus 1 bits of the MSB group and being assimilated and the Nth bit position to produce the final sum ok. So, that means the in the Carry Skip Adder if I got the that means, in the LSB position, if I got 1. So, at the time it will has that means, it has the longest path to be covered ok but based on that we are basically trying to develop this Variable Block Adder ok.

(Refer Slide Time: 15:54)



So, in the Variable Block Adder what it says?

(Refer Slide Time: 15:57)


The idea behind Variable Block Adder is to minimize the longest critical path in the carry chain of Carry Skip Adder, while allowing the groups to take different sizes. Such optimization in general does not result in an enhanced complexity as compared to the Carry Skip Adder. A carry-chain of a 32-bit Variable Block Adder is shown. The first and the last blocks are smaller, and the intermediate blocks are larger. That compensates for the critical paths originating from the ends by shortening the length of the path used for the carry signal to ripple in the end groups, allowing carry to skip over larger groups in the middle. There are two important consequences of this optimization:

•First, the total delay is reduced as compared to Carry Skip Adder

•Second, the delay dependency is not a linear function of the adder size *N* as in Carry Skip Adder. This dependency follows a square root function of *N* instead.

It is also possible to extend this approach to multiple levels of carry skips which represents a linear programming problem, that does not yield a closed form solution. The speed of such a multiple-level adder surpasses that of fixed group Carry-Lookahead Adder. It also exhibits the lower area and power consumption while retaining its speed. Variable Block Adder has the lowest energy-delay product as compared to the other adders in its class.

In the Variable Block Adder, what it says that in their in the that means, LSB position if you see it has only lesser number of that means, in the LSB and in the MSB, there are lesser number of groups and in the middle position they are more number of that means, groups.

That means here, we are skipping actually what whenever the carry is generated at the time we are skipping the next thing. So that means, the skipping of this we have to generate the carry in each of these groups right. So and that carry is basically skipping based on the that means, whenever this carry is generated at this group. So, based on the logic, it can skip that particular logic as it has been discussed ok. So, now, whenever if we consider more number of bits to compute the carry over here final carry over that will take time.

So, the to reduce that what we are doing we are taking the minimum number of that means, bit at this and the MSB position and then in the middle, we are considering more number of bits ok. So, that is why you can see the, with this is the C in. So, this is for. So, this is for this particular group it is it can skip, then it is it has been considered more from this particular step. So, it can skip up to this and in this particular that is again more from the P 1. So, it can skip over here. So, in this fashion, it can ripple this is the carry signal path and finally, you can get the final C out signal.

So, this is the that means, the carry is in that means, the difference between this earlier architecture and these architecture is that there we are considering the k fixed to each of the that means, this grouping. But here, the grouping is being dependent or the grouping has been done considering variable of k; where the middle position is considering more and the LSB and MSB position that are considered as a as low as possible ok.

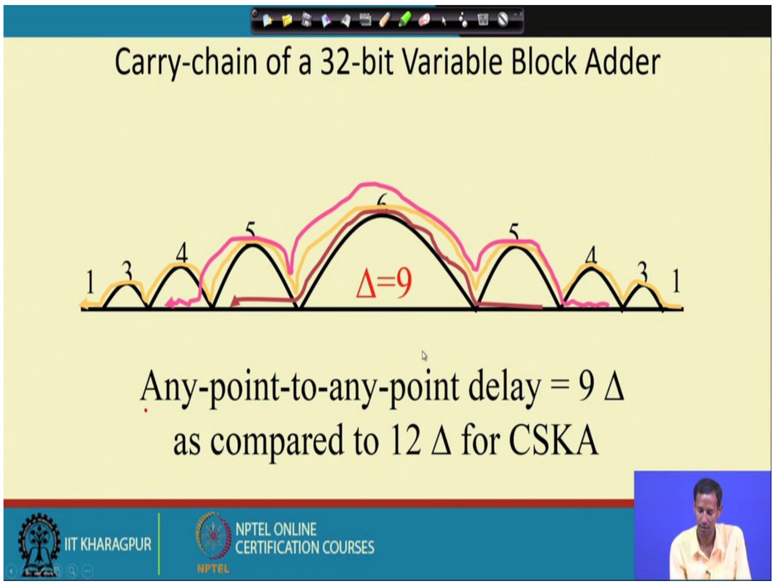
So, the idea behind Variable Block Adder is to minimize the longest critical path in the carry chain of Carry Skip Adder; while allowing the groups to take different sizes. So, what I said that k that can take two different size such optimization in general does not results in an enhanced complexity as compared to the Carry Skip Adder. A carry-chain of 32-bit Variable Block Adder is shown. In the earlier that means, slide the first and the last blocks are smaller and the intermediate blocks are larger.

That compensates for the critical paths originating from the ends by shortening the length of the path used for the carry signals to ripple in the N groups, allowing the carry to skip over larger groups in the middle and there are two important consequences of this optimization ok. The first, the total delay is reduced as compared to Carry Skip Adder and the second is that the delay dependency is not a linear function of adder size N as in Carry Skip Adder.

This dependency follows a square root function of N instead and it is also possible to extend this approach to multiple levels of carry skip which represent a linear programming problem that does not yield a closed form solution. And the speed of such multiple a level added surpasses that of fixed group Carry-Look ahead Adder.

It also exhibits the low power low area and power consumption while retaining its speed. Variable Block Adder has the lowest energy delay product as compared to the others adders in its class ok.

(Refer Slide Time: 20:33)



Carry-chain of a 32-bit Variable Block Adder

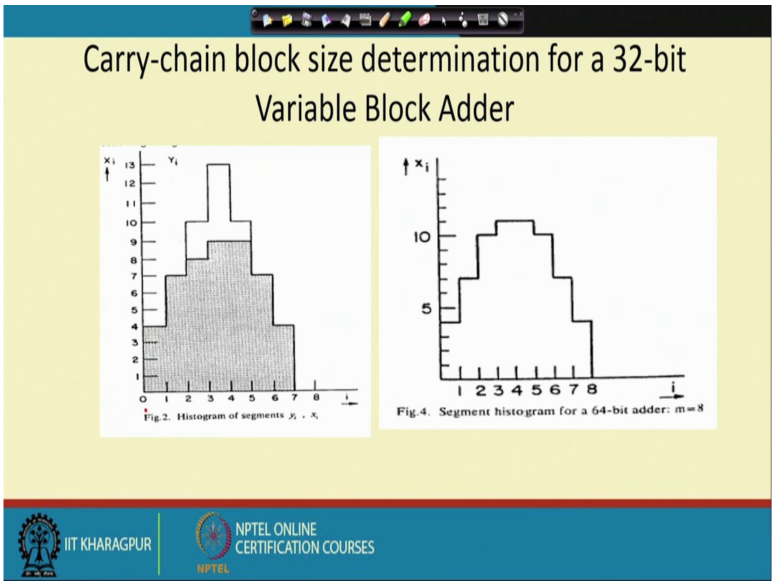
Any-point-to-any-point delay = 9 Δ as compared to 12 Δ for CSKA

So that means, if you see that here for this particular things if I can skip this is for 3 bit this is for 4 bit if I consider this is for 5 in the middle, I can skip more which is 6; then again 5, 4, 3 and then 1 ok. So that means, I can skip the corresponding values in this fashion ok; that means, how we can that means, in the middle position we can skip the more, the point is that we can in the middle.

We can skip the more or we can take the values of k more in the middle position and lower just correspond or just that means, besides of them. That means, in the middle it is more, then the side of that 2 that will be just lower than the middle one; then after that again lower than the previous 1; then something like this we just follow to the LSB and MSB side.

So for any point to point delay here that is if that is 9 delay as compared to twelve delay which is reduce or required for carry skip that k group adder we have designed in the case instead of variable. So, in variable we will get 9 delay; whereas, in Carry Skip Adder we can we get that is 12 delay ok. So, that means, if we consider this particular example; so here at this if you just see that means, here this is basically rippling for this; then for this.

(Refer Slide Time: 22:46)



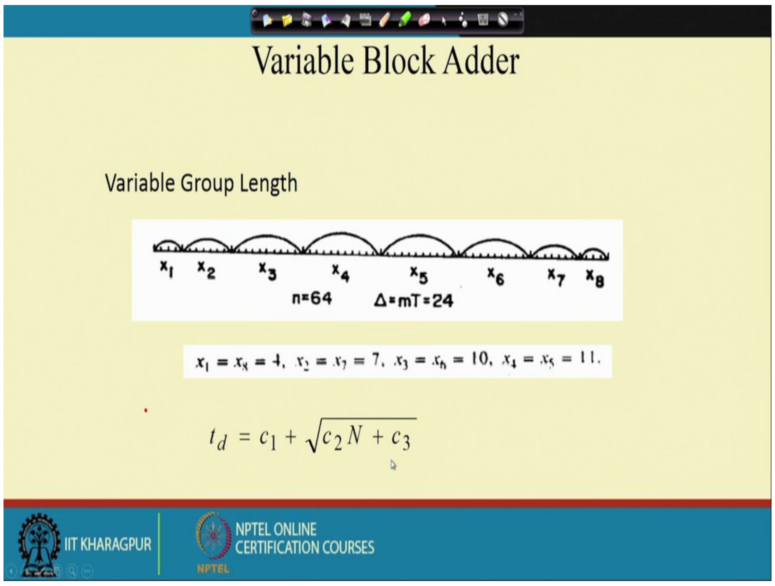
Carry-chain block size determination for a 32-bit Variable Block Adder

Fig.2. Histogram of segments y_i, x_i

Fig.4. Segment histogram for a 64-bit adder: m=8

So, then this is one histogram of how that means, this how this is basically changes; this carry chain block size determination for a 32 bit Variable Block Adder. That means how we can define this k values that we have to do some linear programming for that; that means, for what is the that means, the size of k by which choosing the size of k so that I can get the delay value as low as possible for Variable Block Adder ok.

(Refer Slide Time: 23:30)



Variable Block Adder

Variable Group Length

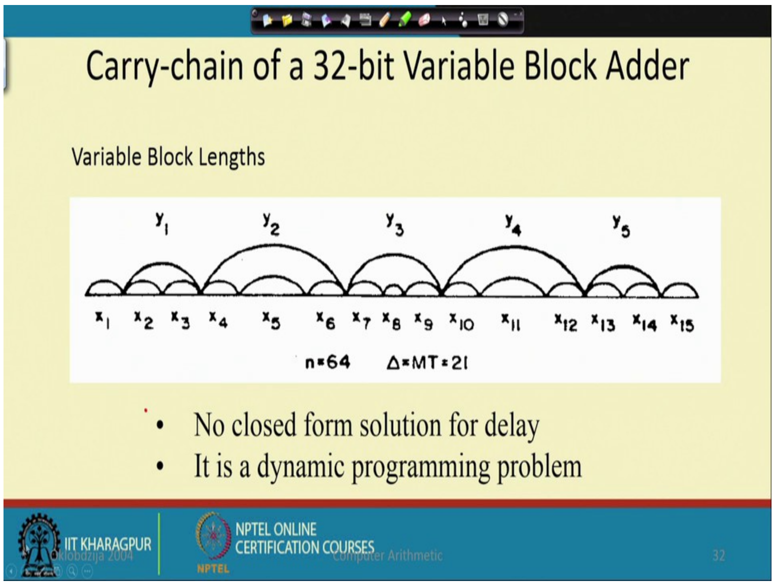
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8
n=64
Δ=mT=24
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

$x_1 = x_8 = 4,\ x_2 = x_7 = 7,\ x_3 = x_6 = 10,\ x_4 = x_5 = 11.$

$$t_d = c_1 + \sqrt{c_2 N + c_3}$$

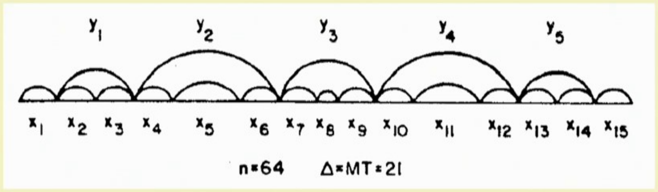
So, if you just see that here, this N equals to 64 ok.

(Refer Slide Time: 23:52)



Carry-chain of a 32-bit Variable Block Adder

Variable Block Lengths



y_1 y_2 y_3 y_4 y_5
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} x_{12} x_{13} x_{14} x_{15}
n=64 Δ=MT=21

- No closed form solution for delay
- It is a dynamic programming problem

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Oklobdzija 2004
Computer Arithmetic
32

And how for x 1 and x 2 they are being selected as 4 and for x 2 and for x 2 and x 7, they are being selected as S 7; x 3 and x 6, they are as 10 and for this x 5 which is the middle of this that is x 4 and x 5 that is selected as 11.

So that means, for the lower that means, this position and this particular position, they are selected as low as possible which is the 4 1 ok. So, the delay is considered or the delay which is the that means, the considered here which is the C 1 plus mod of C 2 N plus C 3; that means, here this is this the group corresponding delay for carry 1. What is the delay then for carry 2; what is the delay for carry 3? What is the delay that are basically considered here for computing the corresponding critical path and here, you see this is the square root the delay is basically not dependent on the linear dependency it is not there it depends on the square root operation of N.

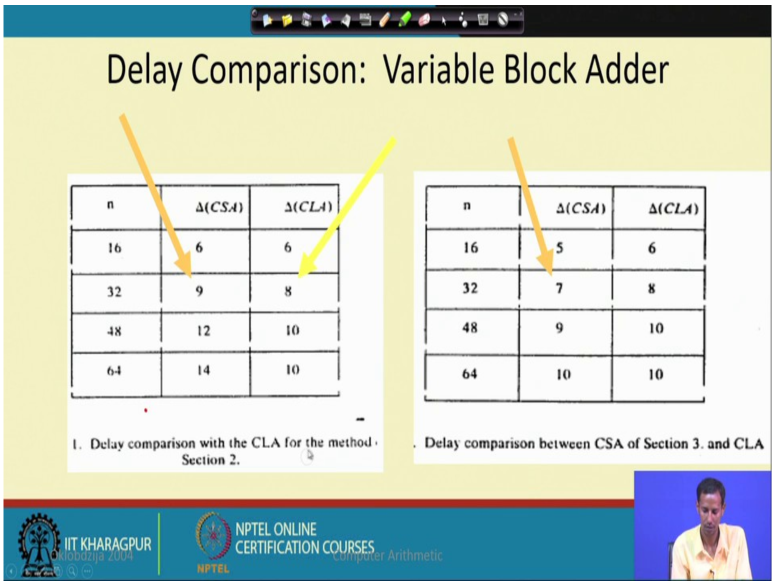
So, that is the beauty of Variable Block Adder and here you see that means, another example where you; that means, I have chosen more number of; that means, group we which is like x 1 to x 15 ok. And then, that means, if I consider I can just make it like 1 2 3 4 5 also I can make or I can do what I can just make another x 1 x 2 x 3 x 4 x 5 x 6 x 7 something like this, means what this is the bigger group and then the bigger group again that can be that means, divided into 2 other group.

So, if I consider this as let us consider this as x 1 then y 1 which is the bigger than this that again can be divided into sub divided into 2 other that means, group which is x 2 and x 3; then y 3 that has been divided into 3 other group which is x 4 x 5 and x 6. Then, y 3 that has been divided into 3 other group that is x 7 x 8 x 9.

So, something like this in this manner we can just take different combination of groups and we can get or we can that means, in which particular combination I am getting the minimum number of delay. So, we will consider that; so that means that as the final one ok.

So, that is why it is a dynamic programming problem.

(Refer Slide Time: 27:03)



Delay Comparison: Variable Block Adder

n	$\Delta(CSA)$	$\Delta(CLA)$
16	6	6
32	9	8
48	12	10
64	14	10

1. Delay comparison with the CLA for the method : Section 2.

n	$\Delta(CSA)$	$\Delta(CLA)$
16	5	6
32	7	8
48	9	10
64	10	10

. Delay comparison between CSA of Section 3. and CLA

Oklobdzija 2004

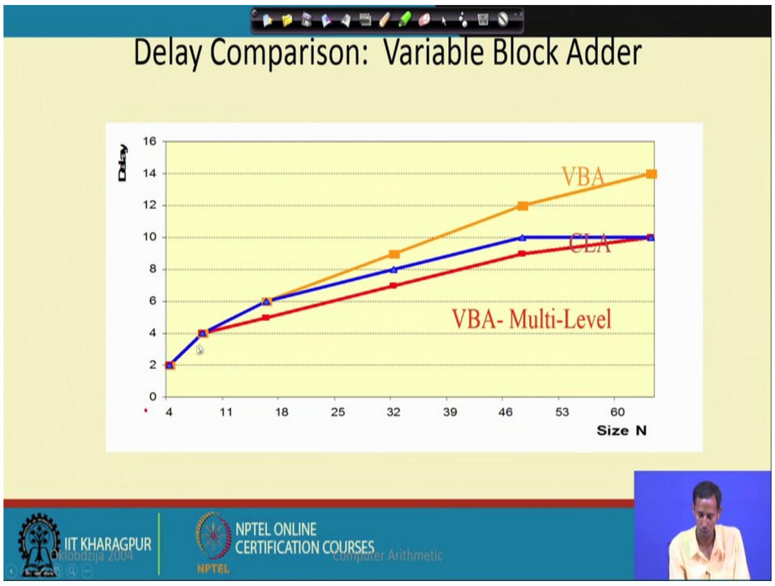
IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES Computer Arithmetic

So, if I compare the delay of Variable Block Adder for this Carry Skip Adder, if I consider this N as 16 and this is for Carry-Lookahead Adder. So, for 32, the delay in Carry Skip Adder that is 9 and in this Carry-Lookahead Adder is 8. So, this is the delay comparison with CLA for the method used in the earlier; that means what we have seen; that means, in this fashion, in this fashion and or in this fashion.

So and if we just do that grouping in this fashion; then that that the delay the for the same thing the delay will be 5 and 6 over here; for 32, it will be 9 and 8; for this it will be 7 and 8; for 42, it will be 12 10; for this it will be 9 and 10; for 64, it will be 14 and 10; it will be 10 10.

So that means, this that means, this particular that means, this grouping is more I can get the delay more reduced in comparison to the architecture or the that means, grouping like this ok.

(Refer Slide Time: 28:44)


Delay Comparison: Variable Block Adder

So, this is that means. So, this is the, that delay comparison we have plotted. So, this is the Variable Block Adder. This is the see the blue one is the CLA and the red one is the, that means, Variable Block Adder Multi-Level, what I discuss in the last. So that means, here you see this is as you increase the size of N; so this has the reduced delay in compared to the other two ok.

So, thank you for today’s class. Again, we will see more adder architecture in the later or in the future classes.

Thank you.