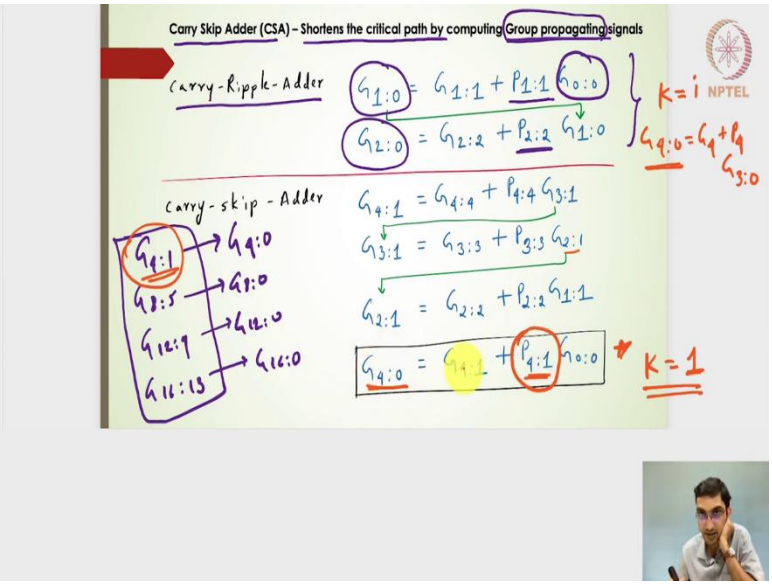
Design and Analysis of VLSI Subsystems
Dr. Madhav Rao
Department of Electronics and Communication Engineering
International Institute of Information Technology, Bangalore

Datapath subsystems - Carry Skip Adder
Lecture - 89
Carry Skip Adder

Hello students, welcome to this lecture on the Data path subsystems where we will focus on the Carry Skip Adder. We had seen earlier the carry ripple adder here we will see take a look at the carry skip adder. We will try to utilize the same propagate and generate signals of course, the group wise propagate and generate signals and then try to design this carry skip adder which turns out to be a little bit faster than that of the carry ripple adder.

(Refer Slide Time: 00:46)



Let us begin with this, what we had seen is the carry ripple adder. Let me take the pointer the carry ripple adder. Here, the way we have utilized the group generate signals is $G_{1:0}$ we generate it from $G_{0:0}$ and then from 1:0 we generated $G_{2:0}$ and then similarly we generated $G_{015:0}$ for a 16-bit adder circuit.

The carry skip adder is nothing but it uses the same the group generate signals, but it also utilizes the group propagate signals and then tries to shorten the critical path. If you remember the carry ripple adder the critical path for generating the S_{16} was coming from

all the generate group generate blocks 15th block, 14th block, 13th block all the way from 1 block.

Here we will try to shorten the critical path by generating the group propagate signals or by utilizing the group propagate signals. In the carry ripple addition, we have not actually utilized the group propagating signals the propagate part was still a bit wise propagate signals.

Here we will try to utilize the group propagating signals to shorten the critical path and, how are we going to do that. We will write the expression for a 16-bit addition we will try to find out $G_{4:1}$, we will try to find out 8:5, we will try to find out 12:9, we will try to find out the 16:13.

Once we have all those things the group of 4 bits. We have done 4 groups of each individual groups having 4 bits. Once we have that then we will try to extract $G_{4:0}$, $G_{8:0}$, $G_{12:0}$, and $G_{16:0}$ and then utilize this 4:0, 8:0, 12:0 and 16:0 to extract the sum bits.

Let us try to understand how we are going to generate this 4:0. That is what I have written here

$$G_{4:1} = G_{4:4} + P_{4:4}G_{3:1}$$

$$G_{3:1} = G_{3:3} + P_{3:3}G_{2:1}$$

$$G_{2:1} = G_{2:2} + P_{2:2}G_{1:1}$$

$$G_{4:0} = G_{4:1} + P_{4:1}G_{0:0}$$

Remember that in a carry ripple ladder the way we generated 4:0 was $G_{4:0}$ was nothing but G_4 plus the bit level P_4 adding with that of 3:0.

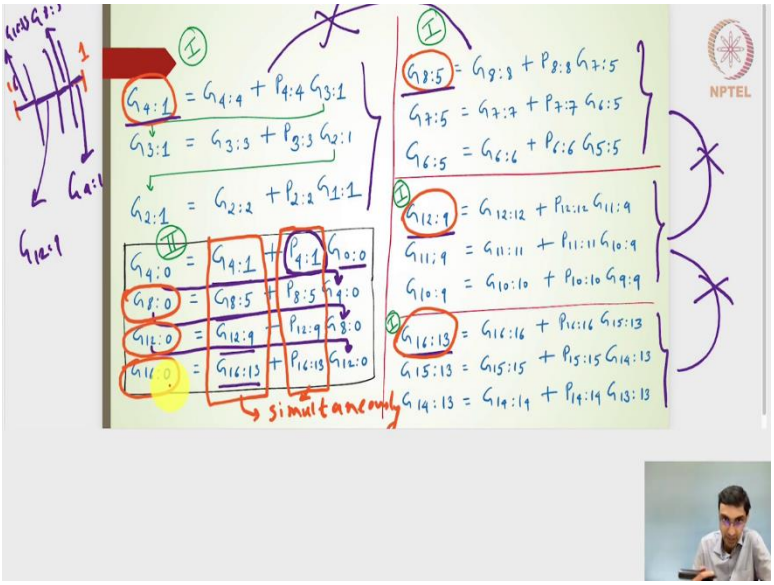
Here the way we are generating it is different $G_{4:0}$ is $G_{4:1}$ that has been generated and $P_{4:1}$ and then $G_{0:0}$. Here in the carry ripple addition we have taken k is nothing but equal to i for the 4th bit, 8th bit, 12th bit and then the 16- bit group generate signals.

Here for the fourth bit 4:0 or the 8:0 or the 12:0 or 16:0 we have taken k value. In this case the k value is nothing but 1 in the subsequent $G_{8:0}$ we will take the k value to be ϕ and for

12:0 we will take k = 9 and then for the 16:0 we will take the k to be nothing but 15 or rather k 16:0 we will take the k value to be nothing but 13.

What we have done is we have utilized a different k value and then try to generate this group generate signals. We have now utilized the group propagating signals in this case and then try to represent $G_{4:0}$ in a different manner. Although for a 4-bit addition we will not be able to see the impact of this particular design.

(Refer Slide Time: 05:15)



Let us try to see for a 16-bit addition how it can be impactful. For a 16-bit addition what we are trying to do is we are trying to generate 4:1 signal, we are trying to generate 8:5, we are trying to generate 12:9 and 16:13 signals simultaneously. What it means is simultaneously means is 4:1 depends on 3:1, 3:1 depends on 2:1, 2:1 depends on 1:1.

If I have a group of 16 bits starting from 1 to 16. We are going to divide or we are going to take it logically divided into 4 such groups. The 1:4, 5:8 and then 9:12 and then 13:16. The 4 groups are generated and simultaneously this is going to give me G 4 is to 1, this is going to give us $G_{8:5}$, this is going to give us $G_{12:9}$ and then this is going to give us the $G_{16:3}$.

The $G_{4:1}$ depends on your $G_{3:1}$ and then it is the bit level 4 and propagate bit level 4 bit and 3:1 depends on $G_{3:3}$ and then $P_{3:3}$ $G_{2:1}$ and then so on. All this individual groups of 4 bits can be simultaneously be generated and that does not have to depend on the other

groups, this does not have to depend on 8 is to 5 or 8 is to 5 does not have to depend on the 4:1.

The 12:9 does not have to depend on 8:5, 16:13 does not have to depend on 12:9. If I can actually logically divide into the 4 groups then individual groups can simultaneously generate 16:13, 12:9, 8:5 and 4:1 signals.

Once I have that $G_{4:1}$, 8 is to 5, 12:9 and 16:13 then we can actually generate $G_{4:0}$, 8:0, 12:0, 16:0 by using this group propagate signals 4:1 from $G_{0:0}$. The once this $G_{4:0}$ is generated we can utilize it to for the next one $G_{4:0}$ to generate 8:0, a once 8:0 is there then we can utilize it for generating 12:0. The 12:0 if it is there then you can utilize it to generate 16:0.

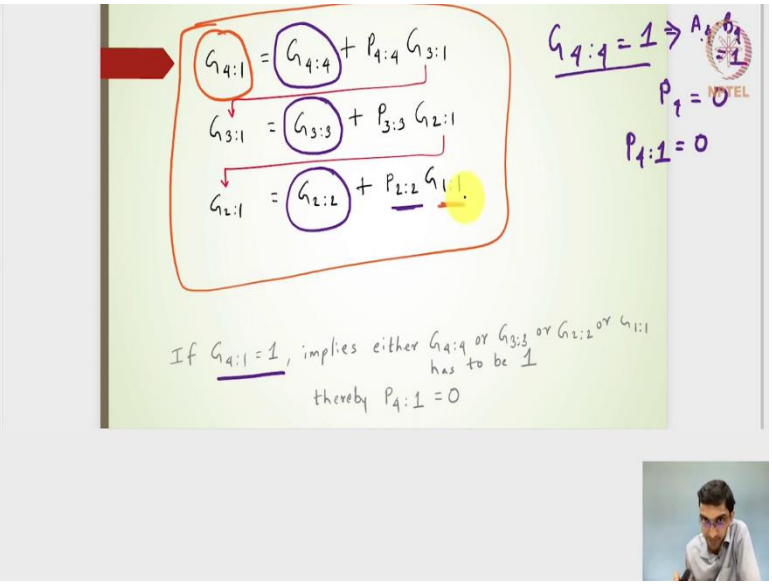
But notice that all this 4:1 to 16:13 are generated simultaneously does not have to wait for the previous the group generate of the lower bits all these things can be generated simultaneously. That is what I am writing down simultaneously and does not have to wait. Once this is available in fact, this $P_{4:1}$, 8:5, 16:13 can also be generated simultaneously.

Once this is done both of them are available and it is just waiting for the group 0 to 0 or 4 to 0 or 8 to 0 or 12 to 0 generate bits. Once these 2 are available with this is also being available at the same time it takes 1 OR gate and then an AND gate to generate 4:0. The 4:0 if it is generated then it takes 1 OR gate and 1 AND gate to generate 8:0 and then so on.

What it means is it actually skips 8:0 can actually skip generating 7:0 or 6:0 or 5:0. In fact, if 4:0 is made available it can directly generate 8:0. The 12:0 can actually skip $G_{11:0}$, $G_{10:9}$ and 9:0. In fact, whenever the $G_{8:0}$ is made available with 1 and 1 OR gate it should be able to generate 12:0.

The $G_{16:0}$ does not have to wait for $G_{15:0}$, 14:0, 13:0 the moment 12:0 is available it can generate a $G_{16:0}$. It skips $G_{15:0}$, 14:0 and 13:0 and directly whenever 12:0 is made available, it then directly generates $G_{16:0}$.

(Refer Slide Time: 09:56)



To visualize let us try to rewrite this expression.

$$G_{4:1} = G_{4:4} + P_{4:4}G_{3:1}$$

$$G_{3:1} = G_{3:3} + P_{3:3}G_{2:1}$$

$$G_{2:1} = G_{2:2} + P_{2:2}G_{1:1}$$

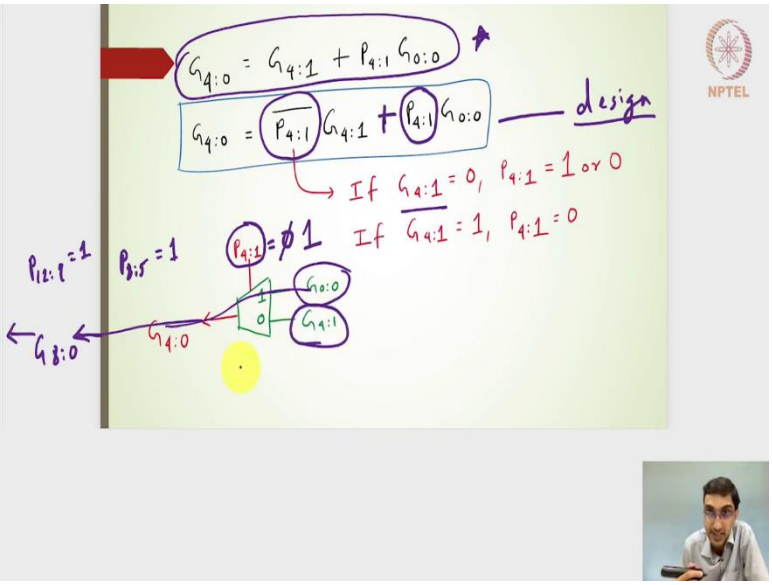
What it means is if I look into this particular expression what it says is, if I want to get $G_{4:1}$ as 1 then either $G_{4:4}$ has to be 1 or $G_{3:3}$ has to be 1 or $G_{2:2}$ has to be 1 or else $G_{1:1}$ has to be 1 and $P_{2:2}$ has to be 1, $P_{3:3}$ has to be 1, $P_{4:4}$ has to be 1 and then thereby I will get $G_{4:1}$ as 1.

Now, if I look into this P_4 $G_{4:1}$ as 1 or if let us say $G_{4:4}$ as 1 in this case. So, that I will be able to generate $G_{4:1}$. The moment the $G_{4:\$}$ is one that means, that my A4 and adding without of B_4 has to be 1, that means A_4 and B_4 bit are 1.

In that sense my P_4 bit has to be 0. The XOR operation of both the bits are 1. My P_4 has to be 0 that means, my $P_{4:1}$ is actually 0 now. The $P_{4:1}$ is nothing but the XOR operations of all P_4 P_3 P_2 and P_1 alright, in that sense it has to be 0.

If I consider this $G_{4:1}$ as to be 1 and if it is 1 because of this the individual generate bits G_4 G_3 G_2 then my $P_{4:1}$ will become 0. That is what I am implying if $G_{4:1}$ is to 1 implies that G_4 or G_3 or G_2 or in this case G_1 has to be 1. If G_1 has to be 1 even in that case P_1 will be 0. The $P_{4:1}$ will actually be 0, that is what I have written.

(Refer Slide Time: 12:41)



Moving ahead. What I am doing is if $G_{4:1}$ is actually 1, we know that $P_{4:1}$ will be definitely be 0. If that is 0, I can actually take $P_{4:1}$, I can virtually represent this particular expression as P 4 is to 1 and then the complement of that,

$$G_{4:0} = G_{4:1} + P_{4:1}G_{0:0}$$

$$G_{4:0} = \overline{P_{4:1}}G_{4:1} + P_{4:1}G_{0:0}$$

If $G_{0:0}$ or rather $G_{4:1}$ is 0 then $P_{4:1}$ can be 1 or 0. It still depends on P 4 is 1 whether it could be 0 or 1. The $G_{4:0}$ if it is $G_{4:1}$ is 0 then it depends on $P_{4:1}$. If it is 0 or 1 it can pass $G_{0:0}$ to 4:1.

If $G_{4:1}$ is actually 1 then that $P_{4:1}$ is definitely 0. I can use this particular complement expression without any change in the logical expression of this. Now if suppose I use this particular expression for designing my circuit.

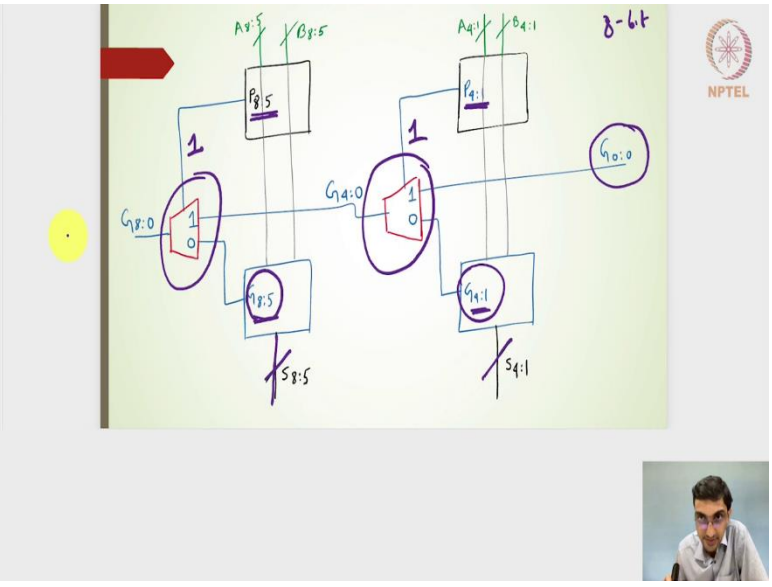
This I am going to use it to design my circuits how am I going to do that. We can utilize this $P_{4:1}$ the complement of that use it as a select line for a multiplexer and then the pass the 2 inputs of $G_{0:0}$ and 4:1.

The $G_{0:0}$ and 4:1. If P 4 is to 1 the select line is 1 then I will pass the $G_{0:0}$ at the output, if $P_{4:1}$ is actually 0 then I will pass $G_{4:1}$ to the output of the multiplexer. This is very important to understand that once I have the group propagating signals $P_{4:1}$ or whether it is 1 or 0. It just passes let us say that if it is 0 in this case, it will pass $G_{4:1}$ to the $G_{4:0}$ output.

If it is 1 here then it passes $G_{0:0}$ into the $G_{4:0}$. Similarly, if I have $P_{8:5}$ simultaneously generated when $P_{4:1}$ is generated and if it is 1 here it will pass this 4:0 to the output here as $G_{8:0}$ and similarly if I have 12:9 as 1 here, it will pass $G_{8:0}$ to the next output of the multiplexer.

The movement $P_{4:1}$, $P_{8:5}$, $P_{12:9}$ is to 1 it can actually skip, it will actually skip generating or the $G_{4:1}$ it will directly take the output from $G_{0:0}$ this will go as 8:0, this will go as 12:0. It can actually skip generating the intermediate generate signals because of the multiplexers.

(Refer Slide Time: 16:04)



That is why it is called as the carry skip adder. To put this particular multiplexer block into the reality, we have 2 blocks now. One is generating $P_{4:1}$ from A_4 is to individual A and B input bits. For an 8-bit addition here we have created 2 such blocks one is generating $P_{4:1}$

the other one is generating $p_{8:5}$ which will go into the select line of the multiplexers. On the lower side, we have 2 blocks one is generating a $G_{4:1}$ another one is generating 8:5. That has been provided that is fed to the one input of the multiplexer, one input of the multiplexer. The other input of the multiplexer is coming from $G_{0:0}$ and output is $G_{4:0}$ which will go on in this particular multiplexer input.

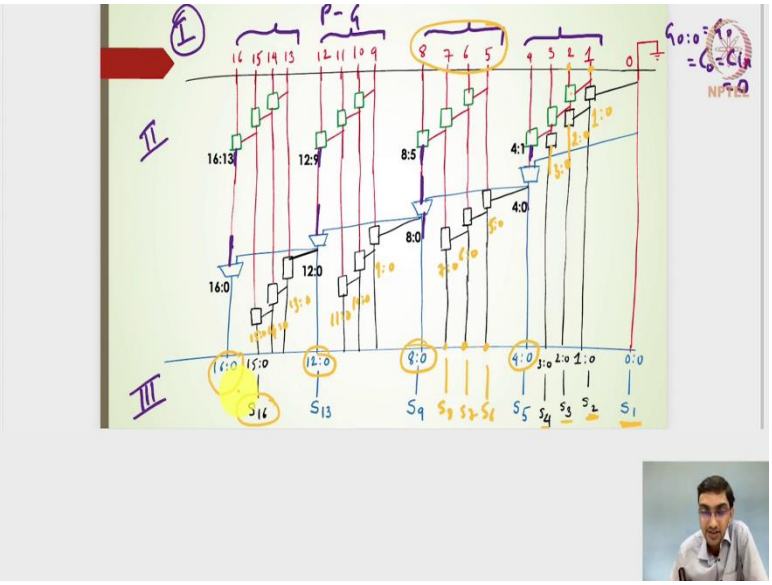
The moment $G_{4:1}$ is 1 here, $G_{4:0}$ will be $G_{4:0}$ it does not have to wait for this to be computed and similarly $P_{8:5}$ if it is 1 here, $G_{4:0}$ which is nothing but $G_{0:0}$ will be passed to the $G_{8:0}$ does not have to wait for $G_{8:5}$ to be computed.

This particular block I have said that it is going to generate the sum bits as 4:1, that means, S_4 S_3 S_2 S_1 and then this particular block is going to generate S_8 S_7 S_6 S_5. This particular multiplexer is going to skip the carry out that is being generated carry out in the sense if I want to generate $G_{8:0}$, it actually skips G_7 is to skips the generation of $G_{7:0}$, 6:0, 5:0. It directly takes $G_{4:0}$ or $G_{8:5}$ based on the group propagate signals.

It may not be advantageous if I have only 4-bit of addition, but it will be advantageous if I have a higher order adder subsystem block design 16 bit edition or 32 bit edition. It will directly skip producing for the 32 bit addition, it will directly skip producing 30:0 or 29:0.

It can actually take if the last group is 1 it can directly skip the generate signals and then they directly take the 28:0 which is actually taken from 24:0 and actually taken from 20:0 and then so on. This particular the carry input of 0:0 can actually be passed to the last bit addition.

(Refer Slide Time: 19:03)



Using this can we now design the PG architecture. The propagate and generate architecture and the first level is nothing but the bitwise, the second one is the group wise where it will generate the group wise generate and group wise propagate bits in this carry skip adder and then the third one is the sum bits generation.

In the first bit what we are assuming is that I have not indicated any of the or rather the AND and the XOR gates here what it means is the XOR and the AND gates are there and it is giving us the bit wise generate and then the propagate signals. The 1 2 3 4 up to 16 represents the bit wise generate and then the propagate signals.

The 0 is grounded represents that the carry input is 0 here. This represents 0:0 which will be nothing but G_0 which is nothing but C_0 or we can also say that it is Cin to be nothing but 0 and this individual blocks here and blocks here, blocks here, blocks here is going to generate the group wise generate and propagate signals. This particular 8 to 5, 12 to 9, 16 to 13 is going to give me 16:13. It could be $G_{16:13}$ and $P_{16:13}$ here 12:9 the output here represents generate $G_{12:9}$ and $P_{12:9}$. Similarly, 8:5 will give me $G_{12:9}$ and $P_{12:8}$ is $P_{8:5}$ and $G_{8:5}$ and this 4 is 1 is going to generate, $G_{4:1}$ and $P_{4:1}$.

All these 3 blocks are at the same level indicates that all of them all the 3 blocks here which is nothing but an AND OR blocks and the XOR operations to generate the propagate signals are there parallely. This block does not have to depend on any of the output of this particular group blocks and then this particular group block does not have to wait for this

particular group of blocks and then so on. That is why it is kind of an independent and all of them if all the signals if 16 to A_1 and B_{16} to B_1 are made available all the output of these blocks are generated simultaneously.

Once they are available simultaneously. This particular line which goes to the multiplexer here the multiplexer takes 3 inputs, one is coming from 0 is to 0 the other one is $G_{4:1}$ and then the third one the select line is nothing but $P_{4:1}$. Based on the select line it will pass the output whether it is coming from this one or the generate $G_{4:1}$ and then it is going to produce 4:0. This 4:0 is going to this particular multiplexer the next group higher order group multiplexer where this particular line indicates 2 inputs again one is a select line, the other one is $G_{8:5}$, this the second input is $G_{4:0}$.

Based on $P_{8:5}$ we will get 8:0 whether it is coming from 4:0 or whether it is coming from $G_{8:5}$. This particular multiplexer output will go to this particular multiplexer input and then this particular line again 2 inputs one is a select line, another one is $G_{12:9}$.

Finally for a 16-bit addition we will have the fourth multiplexer, one input is coming from 12:0 and then the other line which will have one as a select line the other one as $G_{16:13}$. The output is going to be $G_{16:0}$, with this set of 3 the simultaneous blocks here and then this multiplexer. We are going to get 16:0, 12:0, 8:0, 4:0 really really fast, because, it is going to skip generating 3:0 or 7:0 to 11:0 or 15:0 and directly get the 16:0 from the a 12:0 block from the 8:0 block from the 4:0 block.

But here we still need to extract the sum bits, we still need S_2 S_3 S_4, 0:0 is going to give me S_1 because 0:0 with that of the P_1 XOR I will give me the S_1 and P is anyways available here. The S_2 will require the P_2 signal that is available here with that of 1:0.

I need somehow get the 1:0 block. The 1:0 block I can actually have this particular square block which is nothing but AND and OR block this square block is also AND and OR block. The AND and OR block with this one G_1 P_1 and $G_{0:0}$ is going to generate 1:0 here and this block can be along the same level of this particular blocks, the reason is it does not depend on this particular block at all. It has to take the G_1 P_1 signal coming from here and then $G_{0:0}$. The second block here has to wait for 1:0 and then it will generate 2:0 and then this block will generate 3:0.

This one will give me 3:0, 2:0 and 1:0, that I can produce or extract sum 4 sum 3 sum 2. Similarly, for the other group 8:5 we have got 8:0 here, but I need 7:0, I need 6:0, 5:0, that I can produce S_8 sum 8 sum 7 sum 6.

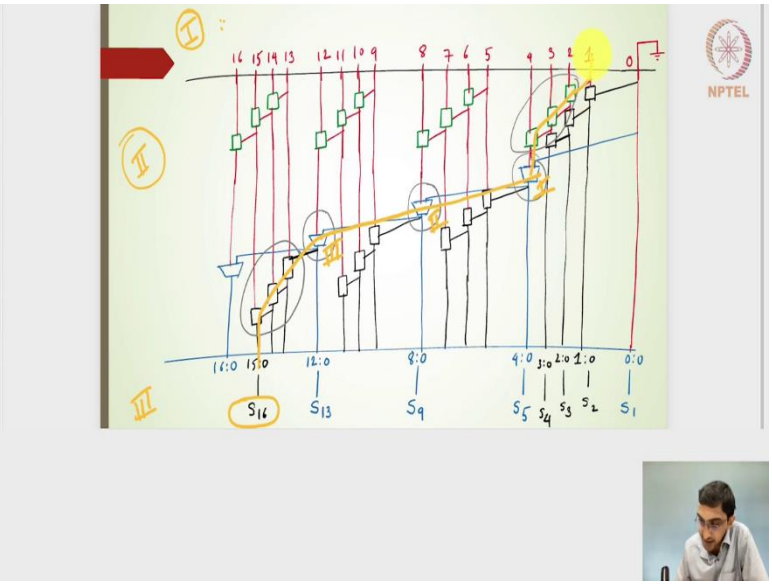
I will produce these 3 blocks here of AND OR gates and this first block will wait for 4 is to 0 whenever the output of the multiplexer at the same time the output of this multiplexer will go to the another multiplexer, but at the same time it will also go into the AND and OR block here.

The AND and OR block here once it produces 5:0, it will go into the next block which will give me 6:0 and then the next block which will give me 7:0. Similarly, I will complete for the other 2 groups generating 9:0, 10:0 and then 11:0 and then the last one the last group I will have 13:0, 14:0 and then 15:0.

Notice that we have multiplexers and then this set of 3 blocks here with the multiplexes is going to give me 16:0, 12:0, 8:0 and 4:0. Only this particular block which are marked in the black lines here are going to give me the output of 3:0, 2:0 and 1:0. Within the individual bits it is inner the group generate bits are generated by this black lines of box and all these square boxes are nothing but AND OR gate representation the multiplexer is a different one.

Once I have this 3:0, 2:0, 1:0 and then 7:0 to 5:0 and 11:0 to 9:0 and then 15:0 to 13:0. I can then produce the sum bits of all this 16 adder output, what should be the delay here. The delay here is considered for the critical path is still the generating the S 16 bit starting from the first bit A_1 B_1 available.

(Refer Slide Time: 27:08)

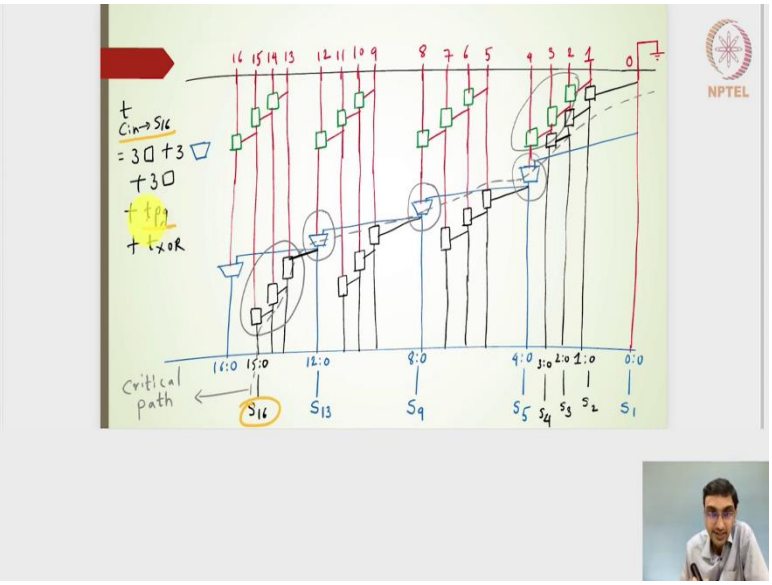


I will have a delay from this particular first stage which is nothing but the bitwise generate and propagate signals the delay due to that. The second one I will have to find the critical path here and then the third one which will be nothing but the XOR operation.

I know the bitwise PG delay and then the XOR delay here. The second one we will have to find out it turns out that the delay is due to this 3 multiplexer output. The multiplexer for this particular group, the multiplexer for the second group here, the multiplexer for the third group and then this is made available for this black boxes. So, as to generate 15:0 and then generate the S_{16}.

My critical path here in the second group will be like this and then this 3 black box does not have to give any inputs to this multiplexer. The multiplexer either gets a input here or it is getting the input here, from this block which is waiting for this block and then this block and then this one.

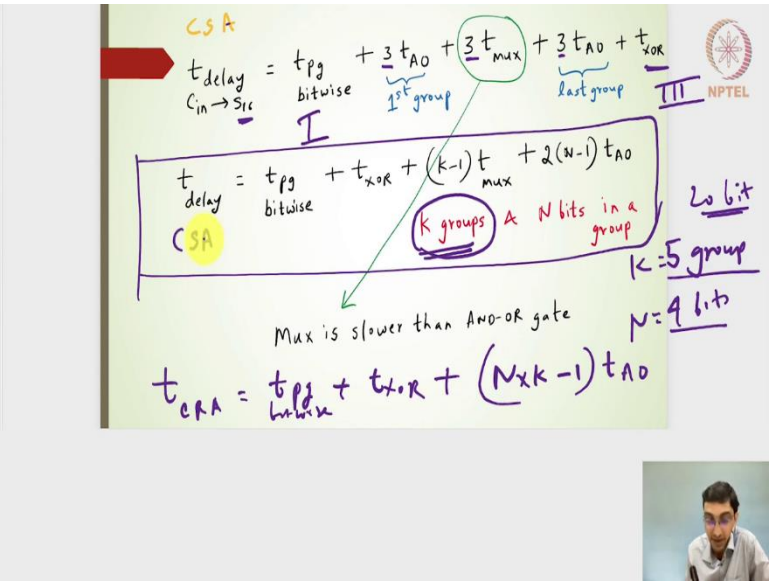
(Refer Slide Time: 28:18)



My critical input or rather the critical path which is going to give me the delay expression for this carry skip adder starting from the C input or whenever the inputs are available to see S_{16} output is nothing but this 3 delay of this 3 boxes plus these 3 multiplexers and then this finally these 3 black boxes and plus of course, the XOR gate and then the PG in the first level in the first stage the bitwise propagate and generate signals delay.

$$t_{Cin \rightarrow S_{16}} = 3\square + 3\square + t_{pg} + t_{XOR}$$

(Refer Slide Time: 28:52)



In a sense the carry skip adder the delay of that for generating S_{16} bit is nothing but the first stage bitwise propagate and generate signals the XOR gates for the final third stage. This is the first state, this is the third stage and in between I will have the first 3 blocks.

$$t_{\substack{delay \\ Cin \rightarrow S_{16}}} = t_{\substack{pg \\ bitwise}} + 3t_{AO} + 3t_{mux} + 3t_{AO} + t_{XOR}$$

In fact, if I can actually do all these are 3 blocks because we have 4 groups and each groups are of 4 bits. If I can actually make k groups I can actually make an asymmetric groups, asymmetric in the sense the groups having different bits or rather I can have k groups.

$$t_{delay} = t_{\substack{pg \\ bitwise}} + t_{XOR} + (K-1)t_{mux} + 2(N-1)t_{AO}$$

If there is a 20-bit addition then I can have 5 groups of 4 bits each, not really asymmetric, but 5 groups with 4 bits each, k is 5 here and N = 4. Then I can have k, I know this could be represented as k minus 1 number of multiplexers delay plus this t_{AO} and then this t_{AO} will make it 2 (N – 1) t_{AO}.

The 4 bits 4 - 1, 3 into 2 and then the number of groups will be k groups. If I consider this particular delay and then compare with that of the carry ripple adder delay it will be same t_{pg} of bitwise and it will be having the same t_{XOR} in the third stage plus, but here it will be for an N bit. The k x N will give me the total number of addition. The k N – k - 1 of t_{AO} will be my answer for the carry ripple adder whereas, here it is nothing but 2 (N – 1) x t_{AO} + k - 1 x t_{mux} and this carry skip adder is likely to be faster than that of the carry ripple adder.