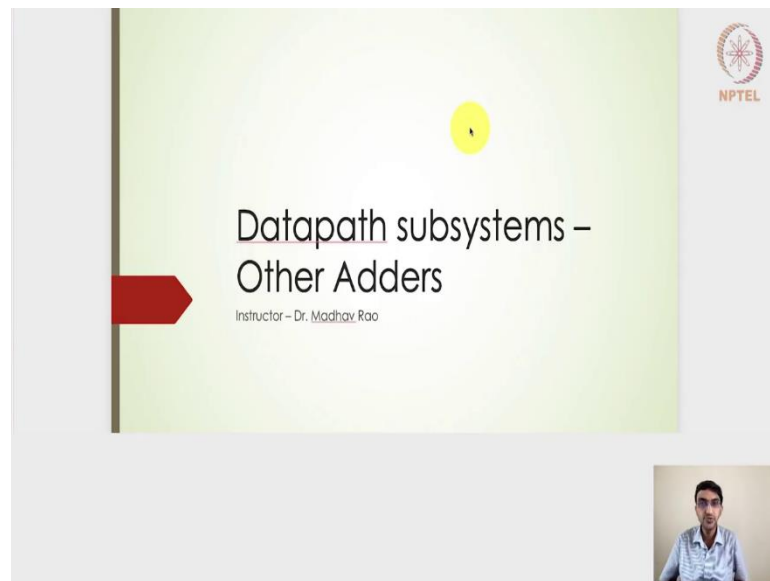


**Design and Analysis of VLSI Subsystems**  
**Dr. Madhav Rao**  
**Department of Electronics and Communication Engineering**  
**International Institute of Information Technology, Bangalore**

**Datapath subsystems – Other Adders**  
**Lecture - 91**  
**Other Adder Subsystems**

(Refer Slide Time: 00:16)



Hello students welcome to this lecture on the Other Adders although I have written it as other adders its mainly we will talk about the carry increment adder. We will just take a revisit the carry increment adder and also have a look at the last adder that will be on the carry select adder that is called as the carry select adder.

(Refer Slide Time: 00:43)

Carry Increment Adder

$$G_{8:0} = G_{8:5} + P_{8:5} G_{4:0}$$

$$G_{7:0} = G_{7:5} + P_{7:5} G_{4:0}$$

$$G_{6:0} = G_{6:5} + P_{6:5} G_{4:0}$$



$$G_{5:0} = G_{5:5} + P_{5:5} G_{4:0}$$

CLA or CSA

$$G_{9:0} = G_{9:5} + P_{9:5} G_{4:0}$$

$$G_{7:0} = G_{7:7} + P_{7:7} G_{6:0}$$

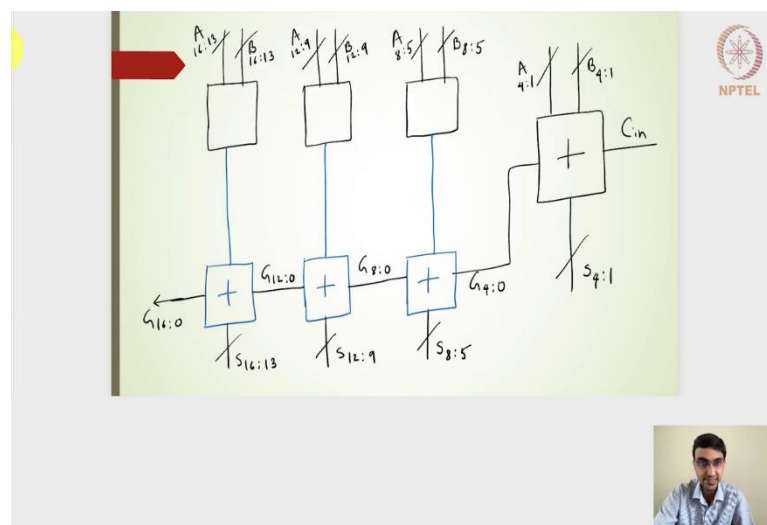
$$G_{6:0} = G_{6:6} + P_{6:6} G_{5:0}$$

$$G_{5:0} = G_{5:5} + P_{5:5} G_{4:0}$$



Let us proceed further just to revisit the carry increment adder and then to understand what is the overall delay. What the carry increment adder does is it utilizes the previous carry out and then simultaneously it generates the  $G_{7:0}$ ,  $6:0$  and  $5:0$  and while the  $G_{8:0}$  is generated. With respect to the difference within the CLA the carry look ahead adder and the CSA which is the carry skip adder.

This is  $G_{7:0}$ ,  $6:0$  and has to wait for the previous carryout. In this particular case the moment of  $4:0$  is made available there is an increment here the OR gate or a plus sign here which represents an OR gate. The moment  $G_{4:0}$  is generated it gets incremented to  $5:0$ ,  $6:0$ ,  $7:0$  and  $8:0$  simultaneously. That is the difference for between the carry increment adder with respect to the previous adders.

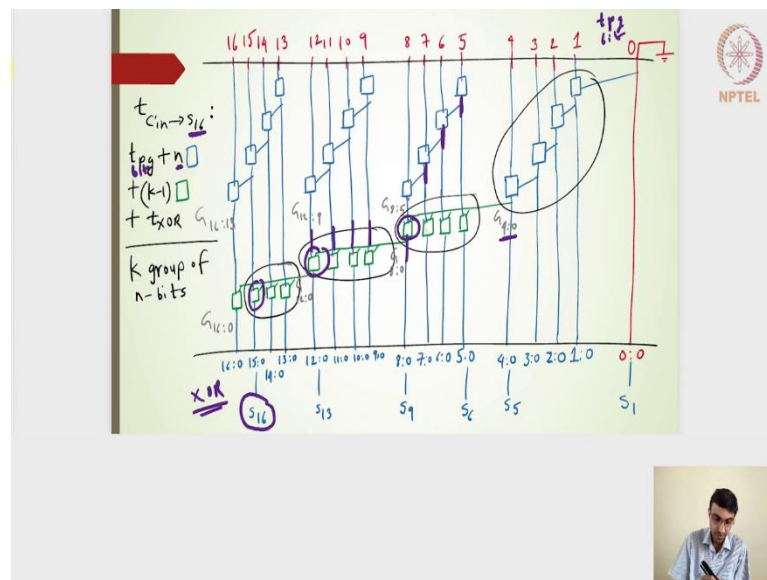
(Refer Slide Time: 01:41)



The block diagram level I think it is pretty trivial. We have the first group of bits generating using the carry ripple edition mode and then afterwards we need to generate the sum bits  $S_{8:5}$ ,  $12:9,16:13$  and so on. Between these two blocks so will be able to this particular block is going to generate it is own group and propagate group generate and propagate signals and then it is going to wait for  $G_{4:0}$ .

The moment 4:0 is available this will all generate  $G_{8:0}$  here and  $G_{7:0}$ ,  $6:0$ ,  $5:0$  at the same time. So, that we will be able to extract the sum bits similarly for the  $12:9$  and  $16:13$  groups also.

(Refer Slide Time: 02:31)



This is the pg level diagram which we had seen in the previous lecture. I will just take my pointer and this particular first 4 bits are nothing but the carry ripple addition. So, as to generate the  $G_{4:0}$  and then what we had said was it will generate  $8:5$ , it will generate  $7:5$ ,  $6:5$  and of course, this will be  $5:5$ .

It is all going to wait for  $G_{4:0}$  the moment it waits all this and an OR gates here are represented as the green boxes, it will made for 40. The moment 4:0 is available then we will generate all of them 8 is to 0,  $7:0$ ,  $6:0$ ,  $5:0$  at the same time.

Similarly, for the other groups also it will  $12:9$ ,  $11:9$ ,  $10:9$ ,  $9:9$  are all made available. The moment  $8:0$  is available then we will generate  $12:0$ ,  $11:0$ ,  $10:0$  and  $9:0$ .

Similarly, for the last group also and to find out the delay we need to find out the S 16 bit. How much time it takes to generate the 16th bit the sum the 16th sum bit starting from the first stage? the first stage will anyways have the  $t_{pg}$  the bitwise delay here and the sum bit it is going to have in the last stage we will have an XOR gate. The delay from the XOR gate, the delay from the  $t_{pg}$  bitwise will be there and in the middle stage we will have this 4, AND and OR blocks delay and then we will have this block delay and then this particular block delay and then this particular block delay.

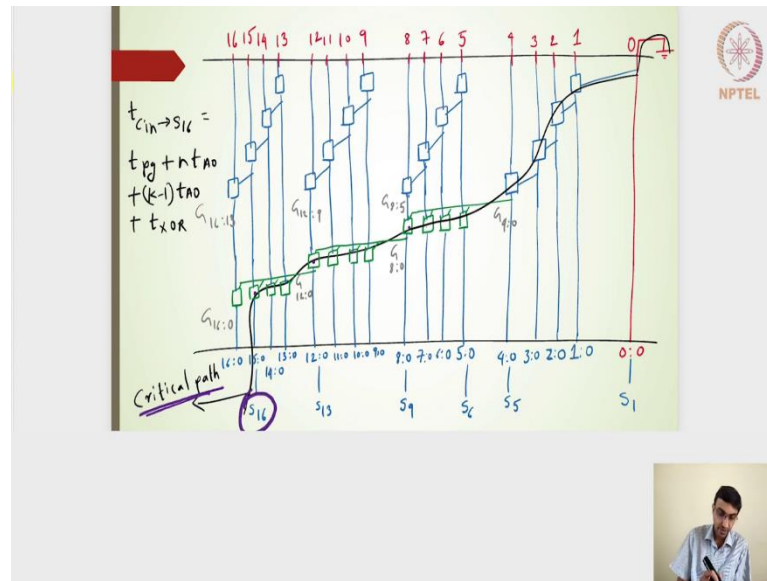
Overall, what we have is from the delay from C input or the inputs whenever it is made available. It can also be instead of Cin it could also be  $A_1$  to  $A_{16}$  or  $B_1$  to  $B_{16}$  and then it generates.

The delay that it takes to generate the S sum 16 bit is nothing, but equal to the  $t_{pg}$  which is nothing, but the bitwise in the first level and then  $t_{XOR}$  is the third stage where it generates or extracts the sum bits and in between the middle stage the second stage will take a delay of the 4 blue boxes here which represent each of these boxes are presents AND an OR gate and then 3 of this green boxes. That is what I have written here instead of 4 I have written an n bits and instead of 3 here I have written  $K - 1$  bit.

$$t_{Cin \rightarrow S_{16}} = t_{pg} + n \square + (k-1) \square + t_{XOR}$$

What it really means is for a K groups for a 16 bits we have divided into the K groups where each group is of 4 bits. In that sense we will have it as K groups of n bits. Instead of 4 here I will have n multiplied by AND and OR gates delay and instead of 3 here I will have  $K - 1$  into the delay of AND an OR gates.

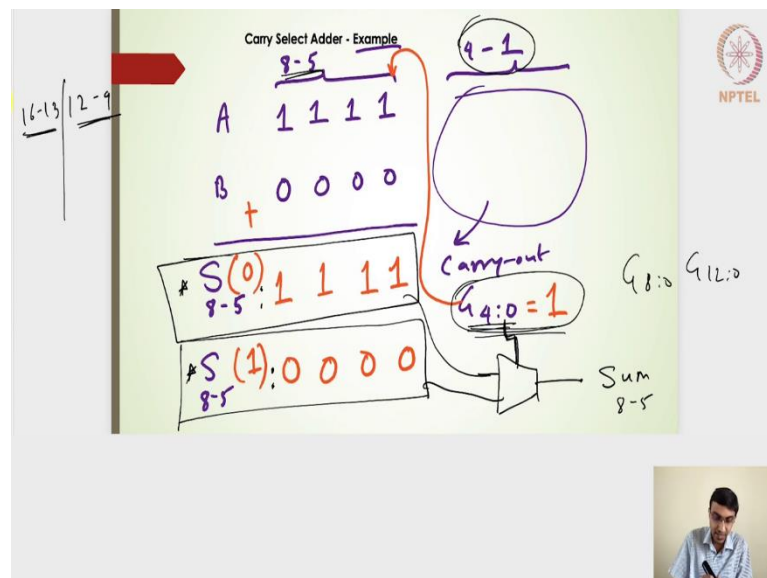
(Refer Slide Time: 05:32)



This is the critical path again starting from the first level to that of the second level where this 4 blue box will be there and then this particular 1 this particular 1 and this particular 1 and then followed by the XOR gate. That will be our critical path so as to generate from the input side to the S<sub>16</sub> bit alright.

$$t_{Cin \rightarrow S_{16}} = t_{pg} + nt_{AO} + (k-1)t_{AO} + t_{XOR}$$

(Refer Slide Time: 05:54)



Let us have a look at the carry select adder, what we had change is we started with a carry ripple adder and then carry look ahead adder carry skip adder and then we also looked into the carry increment adder. Now, let us have a look at the carry select adder.

I am going to take an example first and then we will go into the block level design. Let us say that I have a 4 bit addition. I am not taking a 16 bit or an 8 bit because it is pretty long. Instead of that let us take a very simple example of the 4 bit addition. Let us say that I will have an A and B which is having A will have all 4 bits to be 1, I am taking a very simple case and then B bits is having all of them has to be 0. Let us say that this is my 8 bit to 5 bit addition.

What I am considering is it is in basically an 8 bit addition and here from 4 bit to 1 bit addition whatever is the number it will generate a carryout which is nothing but  $G_{4:0}$ . I am not going to write here whether it is 1 or 0 at this point of time let us say that I will do the addition here right for 8:5. I will get the sum bits here and then I will write it as sum 8:5, irrespective of  $G_{4:0}$  let us say that  $G_{4:0}$  is 0 actually in that sense  $G_{4:0}$  actually has to go and then do the addition here by. But let us say that it is 0 here then I can easily find out what is  $S_{8:5}$  based on the inputs A and B here.

Based on the input A and B here if I do the addition here, I will get 1 I will get 1 here, I will get 1 here, I will get 1 here. This  $S_{8:5}$  is assuming  $G_{4:0}$  is 0, similarly can I now find out  $S_{8:5}$  with an assumption if it is 1. If this is 1 here then this carry out will go here and then this 1 will make this  $1 + 1 = 0$  the carry will be propagated here,  $1 + 1 = 0$  the carry will be propagated here,  $1 + 1 = 0$  the carry will be propagated here,  $1 + 1 = 0$  the carry will be propagated here,  $1 + 1 = 0$  the carry will be propagated here,  $1 + 1 = 0$ .

Of course,  $G_{8:0}$  will be 1, but if I look into this closely. I now have the sum bits which is quite different all of them 1 and all of them to be 0. It is actually based on the value of  $G_{4:0}$  based on the carry out coming from the lower bit groups in this case it is 4:1.

Similarly, I can do it for the 12:9 bit I can also do it for 16:13 bit also. The advantage here is without caring for  $G_{4:0}$  in this particular case can I do the addition and then have the sum generated can I get the sums generated here. Then finally, take a call I have the sum bits are already generated. By assuming the carry input coming to this particular group to be 0 generate the sum bits and also generate the sum bits assuming that the carry input coming to this particular group to be 1.

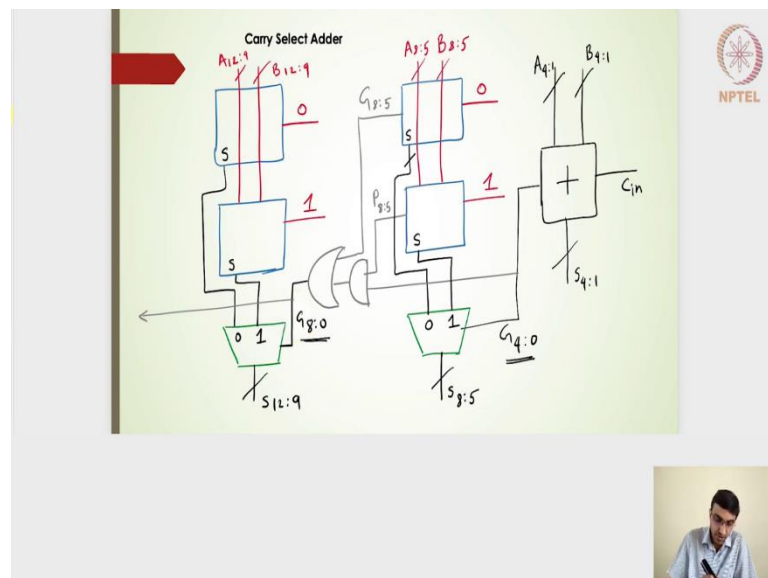
I will have 2 sum bits generated and then this will go into a multiplexer. This goes to a multiplexer let us say and the multiplexer select line will be whenever the  $G_{4:0}$  is created I will give it as a select line and that will be the output of 8:5.

The advantage here is I am actually creating the sum bits for both assuming that the carry input is 0, assuming that the carry input is 1 and both of them will be made available much much earlier. The moment  $G_{4:0}$  is made available I will get the sum bits.

It does not have to really depend on the ripple or the carry out to be rippled from the first bit to the last bit. In fact, the sum bits are generated simultaneously for all the groups 16:13, 12:9, 8:5 it is kind of generated and it is kept the moment the carry out is generated it is going to pass through the sum bits. The delay will be very very less. In fact, the delay will be limited by this carry out signal to be generated.

The moment  $G_{4:0}$  is generated or  $G_{8:0}$  is generated or  $G_{12:0}$  is generated we will get the sum bits. Now, let us have a close look at this with our understanding of that particular example.

(Refer Slide Time: 11:01)



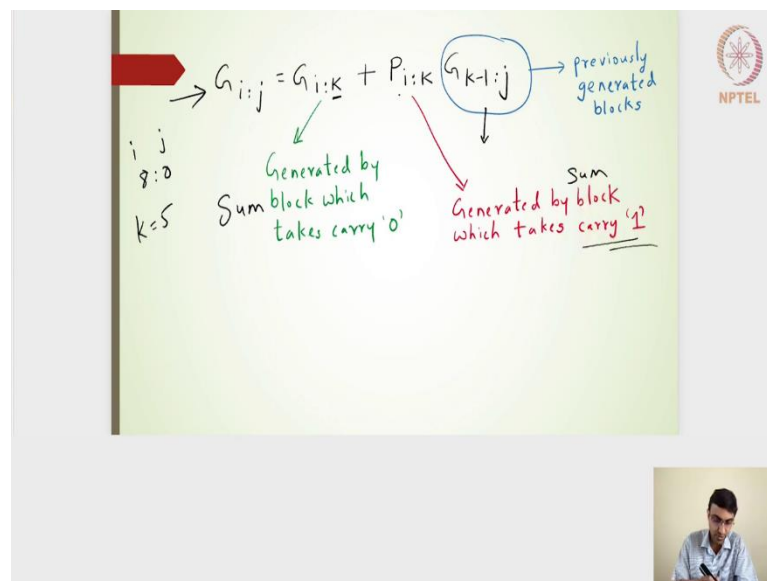
This is the block diagram of the carry select adder, here what we are doing is we have two sum blocks here and then the output is giving to the multiplexers. The 1 sum block assumes that the carry input is 0, the other sum block assumes that the carry input is 1 and then the sum that has been generated this basically the 4 bits here because it is 8:5.

It is actually the sum bits is 4 bits which is given to a multiplexer. In fact, this multiplexer will also be a 4 multiplexers assuming that there are bits to be generated. The output of the 4 bit multiplexes will be  $S_{8:5}$  and this sum bits coming from assuming that coming from this particular block which assumes that the carry input is 1. It is also given to do the multiplexer second input, the select line is given by the whenever this carry out a  $G_{4:0}$  is generated and 8:5 is to generated the moment  $G_{4:0}$  whether it is 1 or 0. It passes this particular sum or it passes this particular sum as the sum 8:5. Similarly, for the next block of 12:9 and the B 12:9 I will have 2 blocks. The 1 block generates the sum bits assuming the carry input is 0 the other block generates the sum bits assuming the carry input is 1 it goes to 4 multiplexes and then based on the carry input 8:0 we will get the real sums.

Similarly, we can continue it for the higher order adder subsystems and notice that for  $G_{8:0}$  and then for this  $G_{8:0}$  or  $G_{12:0}$  or  $G_{16:0}$  that has to be generated it actually uses the  $G_{8:5}$  and  $P_{8:5}$ . This is something we had already seen in our previous 3 adders the carry skip adder, carry propagate look ahead adder and then the carry increment adder.

It uses the same logic to generate and now it uses the same AND an OR logic to generate the  $G_{8:0}$ . The  $G_{8:5}$  is generated within this particular group of bits  $P_{8:5}$  generated from this particular group of bits and then it is fed to the AND an OR gate so as to generate the 8:0.

(Refer Slide Time: 13:26)



Let us do I mean how do we actually do this. This is the expression,

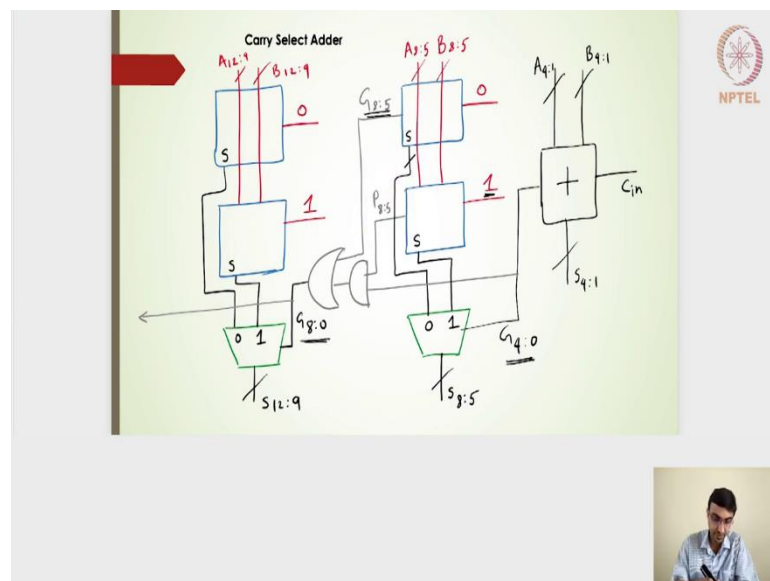


$$G_{i:j} = G_{i:k} + P_{i:k}G_{k-1:j}$$

If it is considered to be 8:0 or something like that so this  $i = 8$  and  $j = 0$ . What we are assuming is  $K$  is something like a you know the bit number 5. It uses  $G_{i:5}$  and then  $P_{i:5}$  has to be created and then 4:0 has to be created. The moment 4:0 is created this is anyways made available by the individual subgroups.

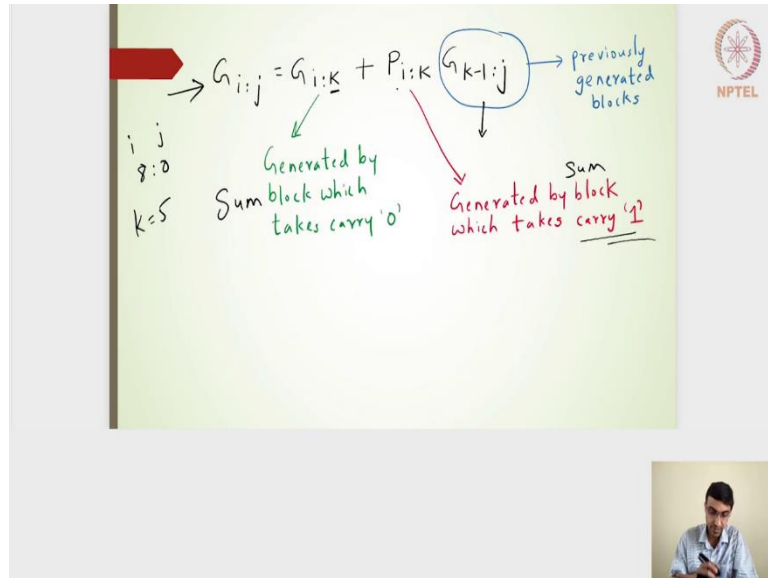
Remember that  $G_{i:k}$  and  $P_{i:k}$  in this case it will be 8:5,  $P_{8:5}$  are actually generated by the different blocks. These are generated by different blocks which takes the generated by the sum block. This is basically sum block which takes the carry 0 and then this is generated the propagate group, propagate signals are generated by the sum block which takes the carry 1.

(Refer Slide Time: 14:39)



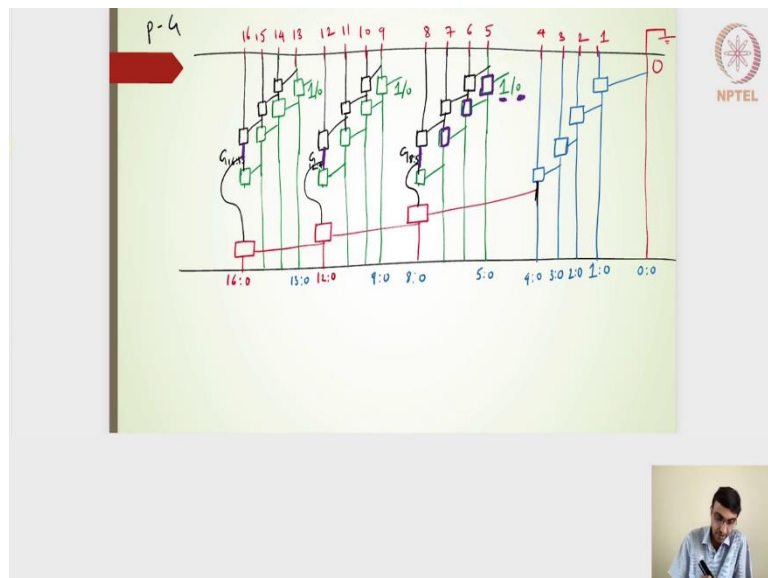
Just to know which block is doing what so whichever is which takes in the carry 0 that block is generating the group generate bit and whichever in is taking 1 that is going to generate the group propagate signals.

(Refer Slide Time: 14:54)



Whichever block is taking the carry 1 that is going to generate the group propagate signal and whichever block is taking the carry 0 it is generating the group generate signals, that is what I wanted to imply in this particular slide.

(Refer Slide Time: 15:09)



Moving ahead, this is the P G level diagram which we have this is the first stage is nothing but we have the bit level bitwise generate and propagate signals. First 4 bits is nothing but the carry ripple addition, 1, 2, 3, 4. It will give me  $G_{4:0}$ . After that let us I will take a look at this particular black boxes here this is nothing but AND an OR gates this is nothing but AND an OR gates and this is nothing but AND an OR gates.

All these 3 black boxes in this 3 group are actually generating  $G_{8:5}$  this is generating 12:9 this is generating 16:13. So, as to generate the  $G_{8:0}$ , 12:0 and 16:0. The moment as 8:0 or 16:0 or 12 is generated it actually helps in you know selecting it.

It goes into the select line of the multiplexer and then we can easily extract the sum bits. Note here that it is basically the sum bits which goes into the multiplexers and it is not the generate signals which goes to the multiplexers or the inputs of the generate signals or the propagate signal that is going to the multiplexer.

The multiplexer actually gets the input as the sum bits, that is why you will notice that in the multiplexer which we will draw now in the next line it will not come in the second stage of the design. It will actually come in the third state of the design because that is generating the sum bits, that is about this 3 black boxes here the 3 black boxes are generating basically the G the individual groups generate and propagate signal.

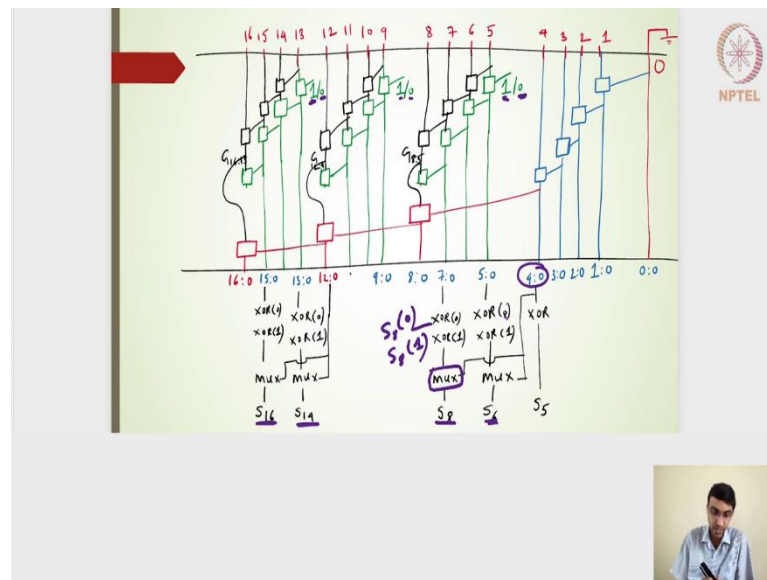
The  $G_{8:5}$  as well as  $P_{8:5}$  here also  $G_{12:9}$  as well as  $P_{12:9}$  here also  $G_{16:13}$  as well as  $P_{16:13}$  and it goes into this red box AND an OR gates. So, as to generate 8:0, 12:0 and 16:0. The moment 8:0 is made available it goes into this particular stage where the XOR gates will be there where the multiplexes will be there, that it will be able to extract the sum bits.

That is about this red box and as well as the black boxes these green boxes are utilized so as to generate the sum bits. Now, I will tell this particular green boxes it takes in 1 as well as 0. This green box there will be two of them 1 takes in the input of 0 carry input as 1 another 1 takes in the carry input of 0. Each of this box here are basically it is a pair of box 1 takes in an input of 0 and the other 1 also takes in an input of 0 or 1.

The 1 and 0 and it creates  $G_5$  it creates  $G_{6:5}$  it creates 7:5 it creates a 8:5, but because the input is already provided this is basically  $G_{8:0}$ , 5:0, 6:0, 7:0, 8:0 and these will be utilized to generate the sum because I am assuming the carry input to be 1 or 0 it is need not have to wait for  $G_{4:0}$ . In this particular stage if I am assuming to be 1 then what is the sum bits, if I am assuming to be the 0 what should be the sum bit?

In fact, in this particular second stage I will actually create 5:0 two  $G_{5:0}$  two  $G_{6:0}$ , 7:0 two such of two a pair of them by taking the input as 1 and 0 moving ahead.

(Refer Slide Time: 19:02)



That is what it is going to create I will have 5:0, 7:0, 9:0. The 10:0, 11:0, 13:0 a pair of them and if it goes to the XOR gate, that I will get an XOR gate with this 0 and XOR gate with this 1, I am going to get the sum of the 7th bit, now rather 7:0 will give me 8 bit with the carry input of 0 the sum 8 bit with the carry input of 1.

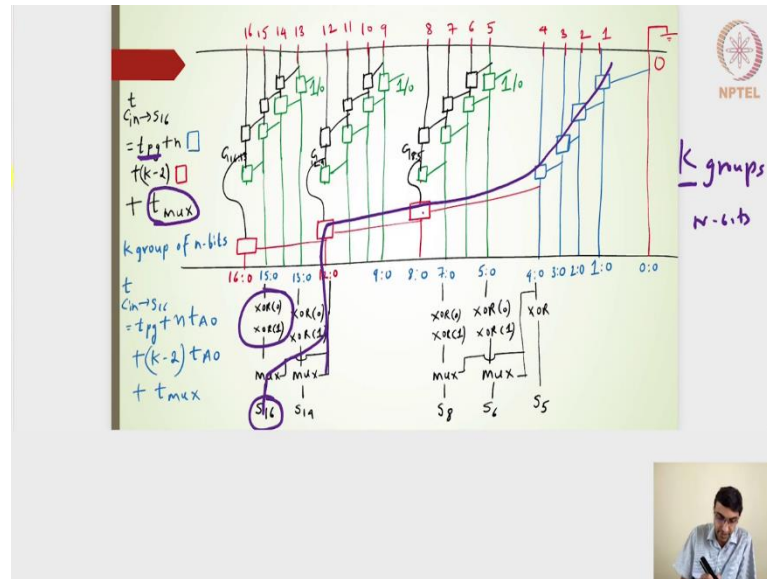
Now, that goes to the multiplexers here to generate the real sum bit where the select line is coming from 4:0. This is about the sum 8 bit. Similarly, I should be able to get the  $S_6$  bit where XOR with at the 0 bit XOR with the 1 carry input is going to create the sum of  $S_6$  with 0 bit with considering 0 and S of 6 bit with considering 1 as an in carry input bit that goes to the multiplexer the select line of the multiplexer is just 4:0 which will generate  $S_6$ .

Similarly, if I consider  $S_{16}$  then based on 12:0 which will go to the select line of this multiplexer. The two inputs to the multiplexer is XOR with this particular 0 or whatever is the sum created and then XOR output of the XOR which is nothing but the sum created with this 1. I will have that goes to the multiplexer and then the output of the multiplexer will be  $S_{16}$ . For generating the sum bits we need this particular green boxes in each of the groups.

Considering 1 or 0 it will be able to generate a pair of  $G_{8:0}$ , 7:0, 5:0 6:0 and or rather 7:0, 6:0 and 5:0 and then that will be provided to the that will be given to the XOR gates to generate the sum bits and then that sum bits the pair of the sum bits will be going into the multiplexer inputs.

The select lines will of course, come from the real  $G_{4,0}$  and then the select lines here also will come from the real 12:0 whenever it is generated and then it is able to generate the right sum bits. I am not populated all the sum bits because it will make it very very difficult to read. I have just used sum of the sum bits  $S_{16}$ ,  $S_{14}$ ,  $S_8$ ,  $S_6$ ,  $S_5$  and then I am saying that all the other sum bits will also follow the similar sequence.

(Refer Slide Time: 21:47)



Now, what should be the delay of this? the delay of this is anyways the first stage the bit level of pg signals. The bit level propagates and generate signals the last level I know it is a little bit different than that of the previous case. This  $t_{pg}$  is nothing, but the bit level and then I am saying that it takes in this particular case n bits. We will have K groups each group is of n bits.

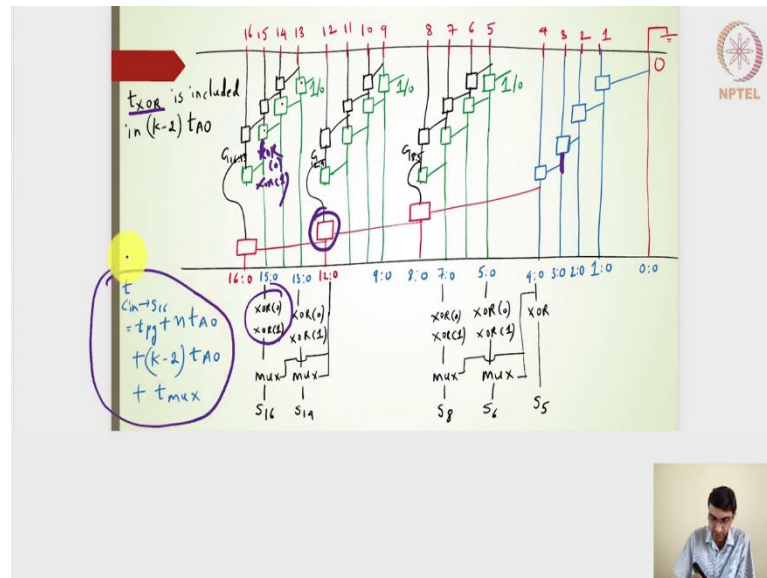
We can say that it is 4 bits or an n bit of the red boxes here followed by we are interested in  $S_{16}$ . The followed by 2 of the red boxes followed by K - 2 here we have made it 4 groups and we are using only 2 red boxes.

$$t_{cin \rightarrow S_{16}} = t_{pg} + n\Box + (k - 2)\Box + t_{mux}$$

The K - 2, 4 - 2 will be 2, 2 red boxes each of these boxes nothing but the AND an OR gate. Here once it comes here it goes to the multiplexers and the outputs are of this the XOR gates considering 0 as the carry input and XOR of 1 considering 1 as the carry input those are made available and then this multiplexer.

The moment 12:0 is generated after this two, after these 4 blocks and then these two red boxes the select line. The delay will also include the  $t_{mux}$  and then  $S_{16}$  will be generated. We have the critical path like this 4 and then this 2 and then this 12 will go here multiplexer and  $S_{16}$ . Here notice that I have not included the XOR bits here or rather this XOR gate both 0 and 1 will be there in parallelly running.

(Refer Slide Time: 23:38)



But we have not included that the reason is, I have said that the  $t_{xor}$  is not included in  $K$ , it is already included in these 2 editions. In fact, the moment this, this and this is done, this is going to create a pair of 15:0.

$$t_{Cin \rightarrow S_{16}} = t_{pg} + nt_{AO} + (k - 2)t_{AO} + t_{mux}$$

This XOR gate will actually happen here whether it is for 0 or whether it is for 1. But my red boxes here it is way down than that of the output of this the green boxes, these 3 green boxes of the same level as that of the output here will appear at the same level XOR the output of this.

But after that it has to go through this and then the red boxes and then it has to go to the MUX. The  $t_{xor}$  gate is not included here because this two cases will actually come here and then the output of the XOR gates the pair of the XOR gates will deliver the output here much earlier than this particular the red box output and that is the reason why the critical path does not include the XOR gate.

The critical part involves these 3 4 or rather 4 blue boxes two red boxes and then the multiplexer. The delay is nothing but this particular expression where it includes the pg bitwise delay and then we have  $4 t_{AO}$  and then  $K - 2 t_{AO}$  in this case it will be  $2 t_{AO}$  and then 1 multiplexer, in terms of making it more generic of  $K$  groups and each group of  $n$  bits, we will have  $n t_{AO}$  and  $K - 2 t_{AO}$ .

Hope this is clear. In this particular lecture what we have seen is we started with the carry increment adder and then we looked into the carry select adder. The carry select adder requires a lot of hardware there will lot of now we assume that the carry input for a particular group will be 1 and 0 and then compute the sum bits. It will require a lot of parallel AND an OR gates and then it is going to generate the pair of sum bits which will be going into the multiplexer groups.

The multiplexer will get a select line that is the select line will be nothing but generated from the carry out of the previous group. Based on the carry out of the previous group then we will have the real sum bits extracted. Overall, the carry select adder turns out to be the really fastest considering than all other the adders. The carry select as well as the carry increment adder turns out to be the fastest among the group of adders which we had seen such as carry ripple adder, carry skip adder, carry look ahead adder. Hope you are able to understand this, thank you for watching this lecture.