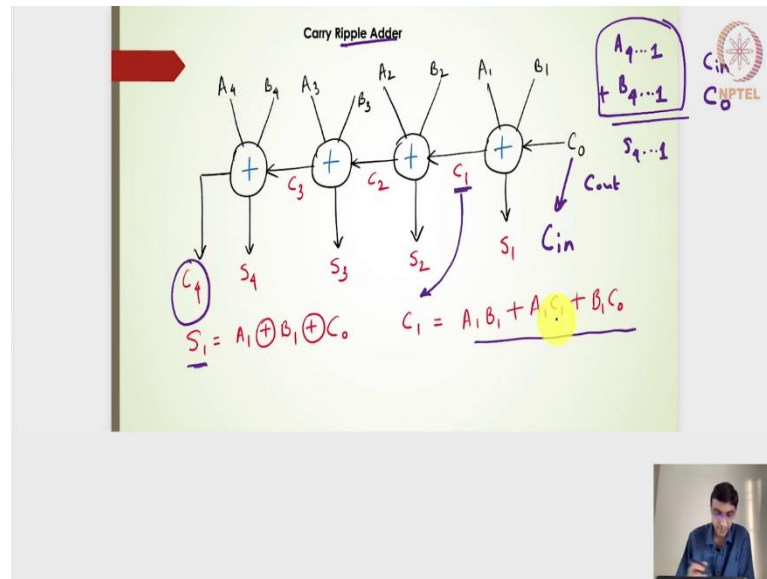


**Design and Analysis of VLSI Subsystems**  
**Dr. Madhav Rao**  
**Department of Electronics and Communication Engineering**  
**International Institute of Information Technology, Bangalore**

**Lecture - 86**  
**Adder-Part2**

(Refer Slide Time: 00:16)



I have written it as  $A_1$  Carry Ripple Adder, the ripple here means that the carry out which is being generated from the individual bit full adder circuit is kind of propagated or rippled to the next bit addition, here it is  $A_1$  very simple example of 4-bit addition.

What it really means is if I have  $A_4$ -bit of  $A_4$  to  $A_{11}$ , where 1 is the LSB bit and 4 is the MSB bit and I need to add it with that of the B input, again 4-bit input, then the addition should give us the sum of 4-bits with  $A_1$  carry out expression.

The  $A_4$ ,  $B_4$  and then there will be  $A_1$  carry out and this addition or this particular adder circuit should also be able to incorporate the Cinput or also the Coutput or also called as the C input to this particular 4-bit addition. Once we understand this, we can easily scale it up to any higher order adder bits. Here at an individual level, at  $A_1$  bit wise level it is the first bit getting you it is  $A_1$  full adder representation of the first bit, the second bit full adder representation.

Similarly, the third bit and then the fourth bit full adder representation. The sum is written here as  $S_1$ ,  $S_2$ ,  $S_3$ ,  $S_4$  and then finally, the out the carry out of the  $A_4$ ,  $B_4$  addition will give

us the  $C_4$  addition which is nothing the  $C_4$  carryout, which is nothing but the carry out of all the 4-bit adder and this also incorporates the carry input, this  $C_0$  can also be written as  $C$  input to this particular 4-bit adders system.

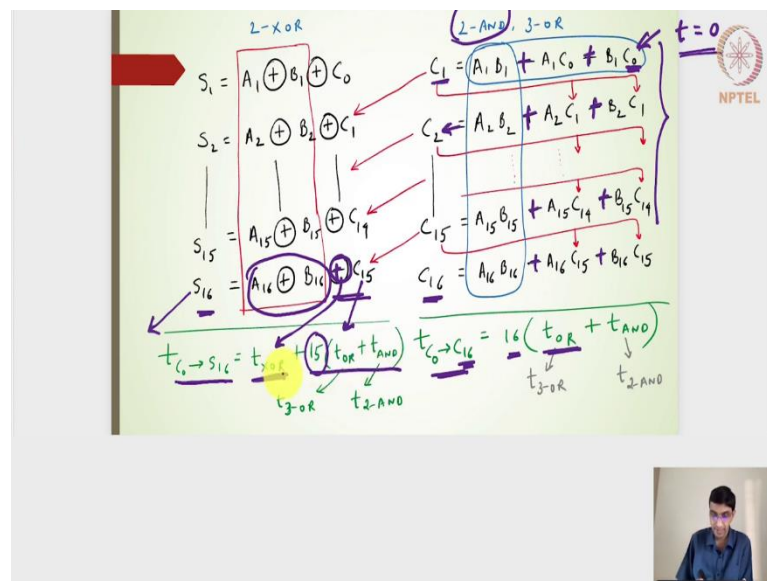
This particular  $C_0$  and then  $A_{11}$  and  $B_1$  should be able to define what is the  $S_1$ .

$$S_1 = A_1 \oplus B_1 \oplus C_0$$

Which is generated which is  $A_1$  carry generated by this first bit of addition is then supplied to the second bit of addition. Then similarly  $C_2$  which is generated is supplied to the  $A_1$  third bit,  $C_3$  which is generated is supplied to the 4-bit addition and then similarly we will get the  $C_4$  as the carry out of the 4-bit addition.

$$C_1 = A_1 B_1 + A_1 C_0 + B_1 C_0$$

(Refer Slide Time: 02:58)



Now, if I want to find out the 16-bit adder right this 16-bit adder is going to generate  $A_1$   $C_{16}$  supplied with  $A_1, B_1, C_0$  and similarly  $A_2, B_2, C_1$  and then similarly  $A_{16}, B_{16}$ .

For  $A_1$  16-bit adder system we will be able to generate the sum 16 or rather the sum  $S_1$  to  $S_{16}$  the carry out  $C_{16}$ , the inputs that are been provided is  $A_1 B_1$  to  $A_{16} B_{16}$ . And then there will be  $A_1 C_0$  that is  $A_1$  carry input to that particular 16-bit addition. These are the inputs and then the outputs are  $S_1$  to  $S_{16}$  and then the carry output of represented as  $C_{16}$ .

The question is what should be the delay for this 16-bit carry ripple adder circuit? let us take  $A_1$  look at it one by one. Let us also assume that all the inputs are available at time  $t = 0$ . All the inputs that means,  $A_1 B_1$  to  $A_{16} B_{16}$  all of them are available at time  $t = 0$ ,  $C_0$  is also available at time  $t = 0$ .

To generate this  $C_1$ , if I want to generate which is nothing but the majority gate of this  $A_1$ ,  $B_1$  and  $C_0$ . It is written as the  $A_1 B_1 + A_1 C_0 + B_1 C_0$  that means, when  $A_1 B_1$  are available easily  $A_1$  and  $C_0$  are also available at  $t = 0$  and  $B_1$  and  $C_0$  are also available at  $t = 0$ ,  $A_1 B_1$  is also available at  $t = 0$ .

If I actually put it into an 2-input AND gate here. I will be able to get the output of  $A_1 B_1$ , AND gate parallel one more 2-input AND gate I will get  $A_1 C_0$ , one more 2-input AND gate will get the output of  $B_1 C_0$ . All these outputs I can give it to  $A_{13}$  input OR gate and then that is going to generate the  $C_1$  bit.

The  $C_1$  bit will take  $A_1$  delay of 2-input AND gate and then the 3-input OR gate, assuming that I will have three of the 2-input AND gates which will be generating the outputs simultaneously and then that will be fed into the 3-input OR gate and then the output will be nothing but the  $C_1$ .

Once I have  $C_1$ , the  $C_1$  is then passed to the  $A_1$  generating the  $C_2$  as well passed to is generating the  $S_2$  alright. Now, I have utilized the delay of 2-input AND gate and then  $A_{13}$  input OR gate. This  $C_1$  is then fed to the  $A_2 C_1$  and then  $B_2 C_1$ , notice is that  $A_2 B_2$ ,  $A_{16}$ ,  $B_{16}$ .

In fact, all this what I have actually annotated with the bounding box here  $A_1 B_1$ ,  $A_2 B_2$ ,  $A_{16} B_{16}$  can be actually be generated if because all of the inputs are available at time  $t$  is equal to 0. This will be generated if I have  $A_1$  lot of parallel 2-input AND gates, then all this outputs of  $A_{16} B_{16}$ ,  $A_{15} B_{15}$ ,  $A_2 B_2$  and  $A_1 B_1$  will be generated at the same time, that will be the delay of 2-input AND gate.

The output of this  $A_2 B_2$  is actually waiting at the input of the 3-input or gate, but it is waiting for  $A_2 C_1$  to be generated and  $B_2 C_1$  to be generated, but  $C_1$  is generated only when we have the output of this 3-input OR gate. The moment  $C_1$  comes in it gets into the AND gate, the output of this AND gate and this particular AND gate will be fed into the 3-input

OR gate and then it generates  $A_1 C_2$ , which will be got here to generate to  $S_3$  the sum 3-bit.

Similarly, it will go here to generate the  $C_3$  bit and then similarly  $C_{15}$  and then similarly  $C_{16}$ . The overall delay from  $C_0$  to  $C_{16}$  is actually limited or its actually depends on the previous carry output bit. What it means is if I want to generate the  $C_{16}$  then it has to depend on the time or the propagation delay for the  $C_{15}$ -bit, before that  $C_{15}$  has to depend on  $C_{14}$ ,  $C_{14}$  has to depend on  $C_{13}$  and so on,  $C_2$  has to depend on  $C_1$ .

Overall, if I look into this particular the carry ripple adder which is generating the carry output for generating the  $C_0$  to  $C_{16}$  for generating the  $C_{16}$  output, it will take OR gate expression, it will take the or expression. It is nothing but the 3-bit OR gate, 3-bit OR gate, 3-bit OR gate, 3-bit OR gate and that will be the 3-bit OR gate is generating the  $C_1$ , the 3-bit OR gate is generating  $C_2$  3-bit OR gate is generating  $C_{15}$   $C_{16}$  and so on.

It depends on the 3-bit OR gate here, as well as the delay also depends on the AND gate here. The moment  $C_1$  is available then it has to wait for the  $A_2 C_1$  to be output to be generated,  $B_2 C_1$  output to be generated and that is fed into the OR gate to generate this the  $C_2$  expression.

For  $A_1 C_2$  to generate the  $C_2$  here, it has to wait for this OR gate, it has to wait for the 2-input AND gate here. The output of that will be fed to  $A_{13}$  bit OR gate and then that will be fed into  $A_{12}$  2-bit AND gate and then that will be fed into the 3-bit or gate.

Even for  $A_1 C_2$  it has to that the overall delay will be twice the 3-bit OR gate plus the 2 AND gate. Similarly for  $C_{16}$ , it is 16 times 3-bit OR gate plus 2-bit AND gate. Now whereas, for  $A_1$  some expression now  $A_1 B_1 C_0$  to  $A_{16} B_{16}$  all are available here at time  $t$  is equal to 0. The XOR operation of  $A_1 B_1$ ,  $A_2$  to  $B_2$   $A_{15} B_{15}$   $A_{16} B_{16}$  and output of it is already available and it is kind of delimited.

The  $S_{16}$  is kind of delimited because delimited by the  $C_{15}$ , whenever the  $C_{15}$  is available it will do an XOR operation and then get the output, because  $A_{16}$  to XOR 16 output is already available and have already reached and it just waits for the  $C_{15}$  to be available. Similarly,  $S_{15}$  will wait for  $C_{14}$  because  $A_{15} B_{15}$  is already the XOR operation is already done.

Assuming that all these XOR gates are parallelly made available and  $A_1 B_1$  to  $A_{16} B_{16}$ , the inputs are already arrived at  $t$  is equal to 0. With that particular condition what should be the delay to generate the  $S_{16}$ -bit, starting from  $C_0$ . What should be the  $S_{16}$ -bit output that is being generated, how much time or the delay it takes.

The delay it takes is nothing but to generate  $C_{15}$  and then one more XOR operation because this is already done. This particular bounded box I have indicated here saying that this is already the moment  $A_1 B_1$  all of them are available at  $t = 0$ , 1 2-bit XOR gate will be generating all this outputs of  $A_1 B_1, A_2 B_2, A_{16} B_{16}$ .

Then it just have to wait for the  $C_{15}$  output, once the  $C_{15}$  output is available then it does an XOR operation and then we will get  $S_{16}$  output. The  $C_{15}$  will be available for when this is generated. This is generated it has to wait for 15 times,

$$t_{C_0 \rightarrow S_{16}} = t_{XOR} + 15(t_{OR} + t_{AND})$$

The  $15(t_{OR} + t_{AND})$  and then one  $A_1 t_{XOR}$  delay will be able to give us the delay for the generating  $S_{16}$ -bit. The one  $t_{XOR}$  means 2-bit XOR gate. whatever is the delay that is this particular XOR gate. The one  $C_{15}$  is available, then it will XOR with the output of  $A_{16} B_{16}$  XOR and then give us. This is my XOR output here and  $C_{15}$  whatever time it takes this is 15 into  $t_{OR}$  plus  $t_{AND}$ .

If I look into this the delay can be actually be expressed in terms of AND and OR gate for generating the  $C_{16}$ -bit or whatever the carry output bit. For the sum bit it is nothing but one XOR and then the 15  $t_{OR}$  and AND gate. This 15 instead of  $A_{15}$  we can also say that for an  $n$ -bit addition it will be  $n$  minus 1 and OR gate and then that of plus the delay of the XOR gate.


(Refer Slide Time: 12:06)

Generate and Propagate Signals

Propagate	A	B	C <sub>in</sub>	S	C <sub>out</sub>	Generate
0	0	0	0	0	0	0
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	0	1	0
1	1	0	0	1	0	0
1	1	0	1	0	1	0
1	1	1	0	0	1	1
1	1	1	1	1	1	1

$P_i = A_i \oplus B_i$

$G_i = A_i B_i$



Let us have  $A_1$  look at it generate and then the propagate signals here. This is the truth table which we have anyways seen this earlier. This particular truth table is anyways seen earlier and then I had also made  $A_1$  very brief definitions of the propagate and generate signal. But once again we will have  $A_1$  look at into it, because then we will be able to use this for designing the adder subsystem blocks for the higher order adder bits.

The generate and propagate signal, for the generate signal the definition is  $A_1$  and  $B$ , the inputs  $A_1$  and  $B$  will be truly generating this carry output independent of the  $C$  input. That is possible only when the carry output is 1 and when  $A_1$  and  $B$  both of them are 1 and irrespective of the  $C$  input being 1 or 0, it actually generates  $A_1$  carryout.

The independent of the  $C$  input, if it generates  $A_1$  carry output signal then the generate signal is 1. That is what the definition says, and that is possible only when  $A_1$  and  $B$  are both of them are 1 alright. Propagate signal its been said that the if it is 1, when the carry out logic is when the carryout is 1 and the carry input is actually 1.

That is possible only in this case, the carry output is 1. The carry input is kind of propagated to 1 and carry input is propagated to carry output as 1 and if I look into the proper definition the propagate  $P$ , it carries if its carryout is true, when there is  $A_1$  carry input.

Going back, the carry output is 1, when there is  $A_1$  carry input. In this particular case it actually generates  $A_1$  here, but this 1 is actually generated by  $A_1$  and  $B$  alright and then not by this carryout, because here carryout is 0 and still it is generating 1.

Whereas here in this particular portion of the combination here the carryout whatever is the carry  $A_1$  carry input it is kind of propagated here, even if it is 0 it is actually propagated to 0 as in the carryout. That is possible only with respect to the  $A_1$  and  $B$  inputs, it is possible only when  $A_1$  and  $B$  one of them is 1.

The propagate signal is actually an XOR operation of  $A_i B_i$  or whatever, in this case if it is  $A_1$  first bit I can write it as  $A_1 \oplus B_1$  or if it is  $A_{19}$  bit it is  $A_i \oplus B_i$ . The generate here if  $A_1$  with respect to the definition of  $A_1$  and  $B$  inputs. The generate bit will be nothing but the and operations of  $A_i B_i$ .

(Refer Slide Time: 15:14)

The slide contains the following text and equations:

- Generate (G) a carry if carry-out is truly independent of carry-in.**
- Propagate (P) a carry if its carry-out is true, when there is a carry-in.**
- Generate and Propagate bits**
- Equations:  $G_i = A_i B_i$  and  $P_i = A_i \oplus B_i$
- Equation:  $S = A \oplus B \oplus C_{in} \rightarrow C_{i-1}$
- Equation:  $C_{out} = AB + BC_{in} + AC_{in}$
- Equation:  $S_i = A_i \oplus B_i \oplus C_{i-1}$
- Equation:  $S_i = P_i \oplus C_{i-1}$
- A diagram on the right shows a vertical stack of two boxes: the top one contains  $S_i$  and the bottom one contains  $C_i$ . A bracket on the left groups these two boxes and is labeled  $P_i$  and  $G_i$ . A yellow highlight is under the  $C_i$  box.

That is what I have mentioned here in the next slide. The generate is the AND operation of  $A_i B_i$ , the propagate is the XOR operation of  $A_i B_i$ . What it really means is if  $P_i$  is 1, that is only when the carryout is kind of propagated from the carry input,  $G_i$  is 1; that means, the  $A_1$  the  $i$ th bit is actually generating  $A_1$  carryout.

With this particular definition, what we know is  $A_1$  sum bit can be expressed as,

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

Then the C input can also be considered as  $C_{i-1}$  and this sum I can also state it as  $S_i$  for the  $i$ th bit  $C_{i-1}$  the carry input  $A_i B_i$  expression. The  $A_i B_i I$  can consider it to be nothing but the  $P_i$ , the propagate bit.

The propagate bit XOR with that of the  $C_{i-1}$  will give me the sum bit. That is how we will be using this bit wise propagate signal to define our sum output and also to that of the carry output.

What we have seen so far is we define the  $P_i$  bit, the propagate bit here, we have defined the  $G_i$  bit here the generate bit wise generate and then-bit wise propagate signal. This we are going to use it to define our sum expression, to define our carry out expression.

The carry out will be nothing but  $C_i$  for an  $i$ th bit the sum for the  $i$ -bit is sum  $S_i$  carry input is nothing but  $C_{i-1}$ . We will use this particular definition to redefine our sum expression and then the carry output expression and try to visualize the different adder circuits.

(Refer Slide Time: 17:06)

Definition of Group  $G_i$  &  $P_i$  still remains the same... Generating Carry-out from the group irrespective of Carry-in to Group, and propagating a carry-out from the group, when there is a carry-in to the group.

Group generate signal

$G_{i-1:0} = C_{i-1}$

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

$G_{1:0} = G_1 + P_1 G_0$

$G_{1:0} = A_1 B_1 + (A_1 \bar{B}_1 + \bar{A}_1 B_1) C_0$

$= A_1(B_1 + \bar{B}_1 C_0) + \bar{A}_1 B_1 C_0$

$= A_1(B_1 + C_0) + \bar{A}_1 B_1 C_0$

3-bit  $C_{in}$   $C_{out}$

NPTEL

Here is  $A_1$  definition of  $A_1$  group wise generate signal. Until now what we have seen is the bit wise  $G_i$  which is nothing but  $A_i$  and  $B_i$ , but here it is  $A_1$  defining  $A_1$  group wise generate signal of  $i-1: 0$ . What it really means is if I am using an adder circuit let us say the 4-bit adder circuit. For  $A_4$ -bit adder circuit, I can actually generate  $A_1$  group wise 3 is to 0.



The  $A_1$  group wise generate signal  $3 : 0$  and then use this for to express the 4-bit addition sum expression and then the carryout expression. What I am saying here is the definition of the group wise  $G_{i:j}$  and  $P_{i:j}$  still remains the same. The  $G_{i:j}$  and  $P_{i:j}$ . This is  $A_1$  group wise now,  $i, j$  are not  $A_1$  individual bit,  $i$  and  $j$  are not same it is  $A_1$  different numbers. Let us say that I can take an example of 3 is to 1 or P of 3 is to 1.

This becomes 3, 2 and 1 bit, 3 to 1 1 bit. The definition still remains the same, saying that generating the carry out from the group irrespective of the carry input to the group. Generating the carryout from the group irrespective the carry input to the group will have to generate  $A_1$  signal as 1.

If I have  $A_1$  block of 3-bits and this is my C input to that particular 3-bits and then this is the output of that particular 3-bits, the definition says if the C output is truly independent of that of C input. In respect to this input if it is generating C output then we can say that this  $G_{3:1}$  is actually 1.

Similarly, propagate signal it is the definition still remains the same. The propagating  $A_1$  carry out from the group when there is  $A_1$  carry input to the group. Similarly, if the C input here is 1 and then C output is kind of this C input is propagated to the C output, we can say that the propagate 3 is to 1 is actually 1. This is the group wise definition of generate and then propagate signals and this particular group wise generating signal of  $G_{i-1:0}$  is nothing but the carry input for the  $i$ th bit.

What it also means is if I can actually do generate the group wise  $i-1$  bit, The  $i-1$  is to 0 group wise generate signal, that is nothing but the carry output of  $i-1$  bit for the group  $i-1$ .

$$G_{i-1:0} = C_{i-1}$$

This is  $A_1$  carry output, it will also act as  $A_1$  carry input to the  $i$ th bit addition. This is what the group wise generate signal is kind of defined. Now, let us take  $A_1$  look at it whether it really makes  $A_1$  any sense here. If I consider,

$$\begin{aligned} G_{1:0} &= G_{1:1} + P_{1:1}G_{0:0} \\ G_{1:0} &= G_1 + P_1G_0 \\ &= A_1B_1 + (A_1\overline{B_1} + \overline{A_1}B_1)C_0 \end{aligned}$$

$$= A_1(B_1 + \overline{B_1}C_0) + \overline{A_1}B_1C_0$$

$$= A_1(B_1 + C_0) + \overline{A_1}B_1C_0$$

(Refer Slide Time: 22:18)

Handwritten derivations on a whiteboard:

$$G_{1:0} = A_1B_1 + A_1C_0 + \overline{A_1}B_1C_0$$

$$= B_1(A_1 + \overline{A_1}C_0) + A_1C_0$$

$$G_{1:0} = B_1A_1 + B_1C_0 + A_1C_0 \quad C_1 = C_{out}$$

$$G_{1:0} = C_1$$

$$S_2 = A_2 \oplus B_2 \oplus C_1$$

$$= A_2 \oplus B_2 \oplus G_{1:0}$$

$$S_2 = P_2 \oplus G_{1:0}$$

$$G_{2:0} = C_2$$

$$S_i = P_i \oplus G_{i-1:0}$$

$$G_{1:0} = A_1B_1 + A_1C_0 + \overline{A_1}B_1C_0$$

$$= B_1(A_1 + \overline{A_1}C_0) + A_1C_0$$

$$G_{1:0} = B_1A_1 + B_1C_0A_1C_0$$

$$G_{1:0} = C_1$$

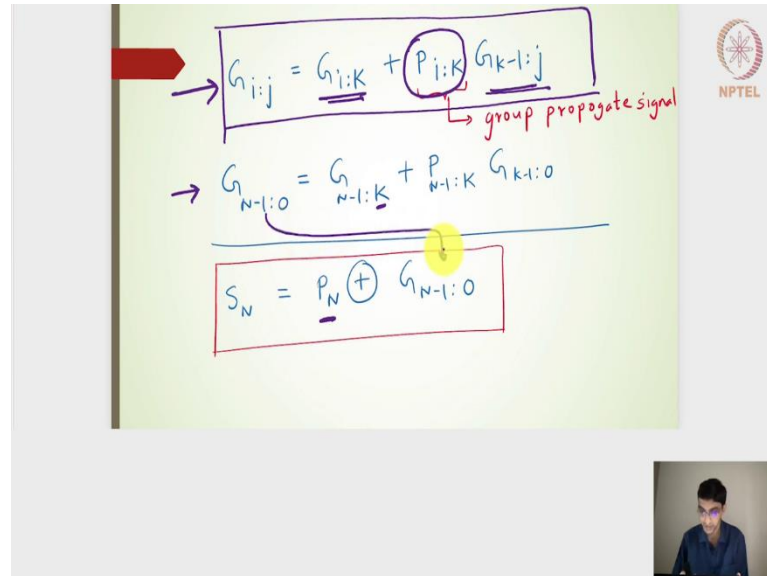
The  $G_{1:0}$  or rather the group generated, the group generating signal for 1:0 is nothing but  $C_1$ . Similarly, if I do the expression  $G_{2:0}$ , turns out to be nothing but the carry two. The sum expression I can actually write it in terms of the group generating signal.

If I want the sum 1 or sum 2 here, once the  $C_1$  is generated we know that it will be nothing but  $A_2 \text{ XOR } B_2 \text{ XOR } C_1$ . Which I can also rewrite it as  $A_2 \text{ XOR } B_2 \text{ XOR } G_{1:0}$ , in terms of the group generating signal and  $A_2$  and  $B_2$  I can actually write it in terms of the propagate bit 2-bit level propagate signal  $P_2 \text{ XOR } G_{1:0}$ .

Similarly, I can actually write  $C_i^{\text{th}}$  bit, it is nothing but the XOR operation of the propagate, bit level propagate signal  $P_i$  and with that of the  $G_{i-1:0}$  alright. If I am generating the C the

sum of the  $i$ th bit, it is nothing but the propagate of the  $i$ th bit XOR with that of the group generate of the previous one. The  $i-1:0$ , hope this is clear.

(Refer Slide Time: 24:32)



This is the overall  $A_1$  broader definition of the group generate signal starting from  $i:j$ .

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

This is  $P_{i:k}$  group propagate signal, this is  $A_1$  group generate signal, this is  $A_1$  group generate signal, this is  $A_1$  group generate signal from the group of the bit wise  $i$ th to  $k$ th bit this is group generate signal from  $k-1$  to  $j$  bit.

This is the propagate which is  $A_1$  group propagate signal from  $i$ th bit to  $k$ th bit. Similarly, we can also define if  $j = 0$  and  $i = N-1$ . It will be nothing but  $N-1$  and then in  $k$  will be  $A_1$  bit which is in between  $N-1:0$ .

$$G_{N-1:0} = G_{N-1:k} + P_{N-1:k} G_{k-1:0}$$

$$S_N = P_N \oplus G_{N-1:0}$$