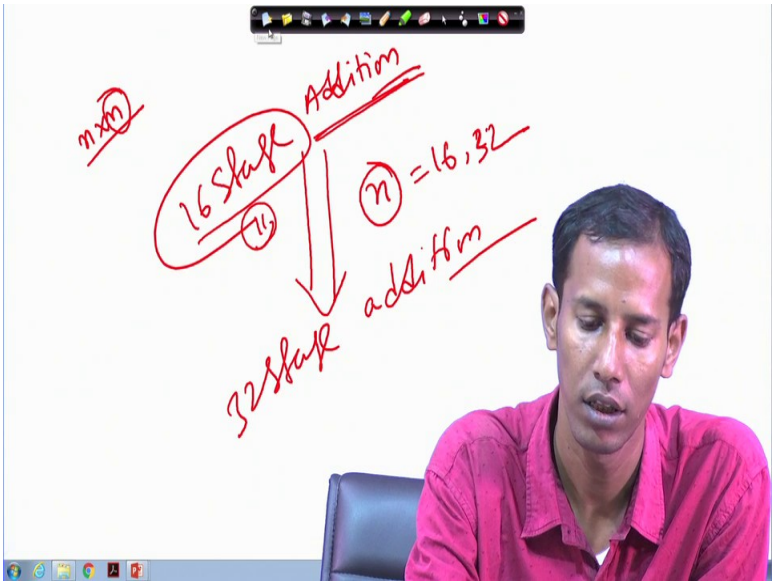
**Architectural Design of Digital Integrated Circuits**
**Prof. Indranil Hatai**
**School of VLSI Technology**
**Indian Institute of Engineering Science and Technology, Shibpur, Howrah**

**Lecture - 30**
**Multiplier Architecture ( Contd. )**

Welcome back to the course on Architectural Design of IC's. So, you we are seeing in the last class we are discussing about the Booth's algorithm for reducing the partial products ok. So that means, what I said that if I consider more number of bits or the word length is more for multiplication operation, so at that time in general multiplication the depth is basically also it becomes more. Means what?

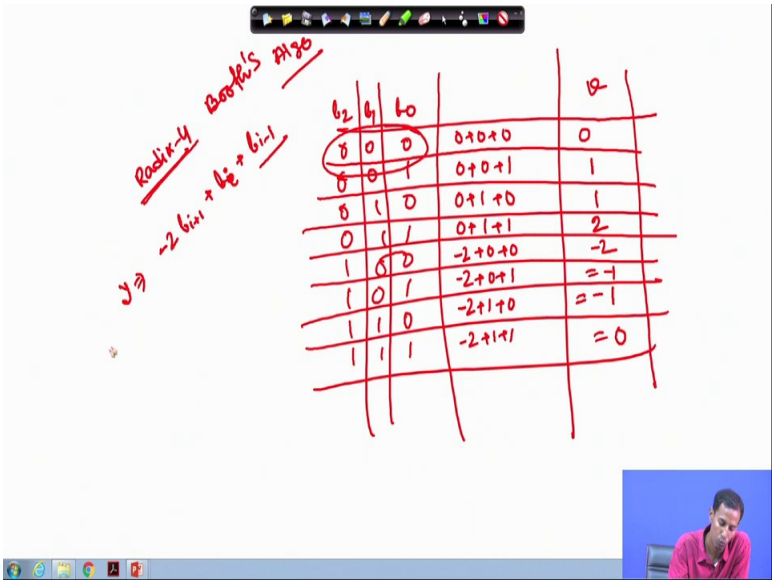
(Refer Slide Time: 00:52)




If I consider n into n bit number of multiplication, so at that time what happens? The depth is also of; that means; this n bit. If n is 16 so; that means, 16 stage of addition operation 16 stage of operation; that means, addition operation I require ok. If n is let us consider 32, so, at that time 32 stage of addition operation I require to perform for the final computation of the multiplication operation. So, that is why what we have done, we have that means, used this Booth's decoding technique to further reduce the number of this stage of this chain addition operation ok.

So, how we have reduced based on this particular algorithm? That means, based on based on this particular equation which is nothing, but this if I choose that Radix-4 Booth's algorithm.

(Refer Slide Time: 02:00)

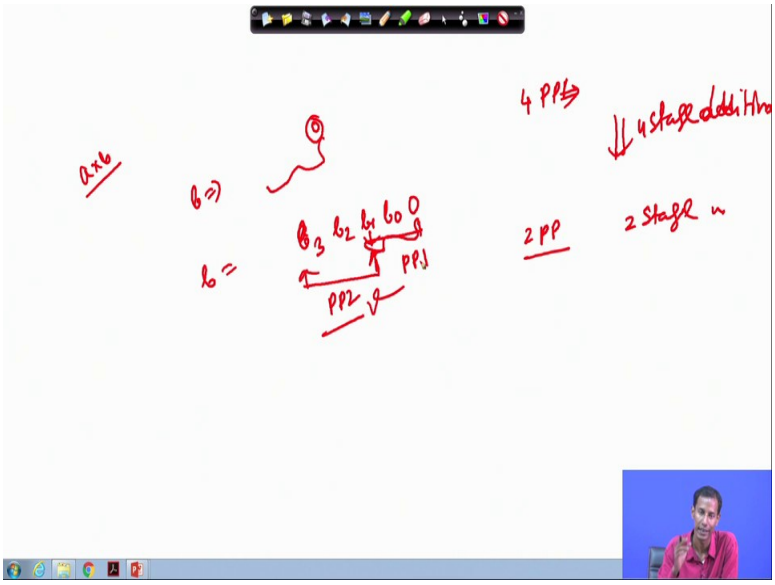



So, at that time the corresponding recoding bit that has been selected as minus 2 into b i plus 1 plus b 0 or bi into bi minus 1 ok.

So, this is for Radix-4 and based on that now so if I; that means I will get what? 3 bit combination for 0 0 0 for 0 0 1; so, this is b 2 this is b 1 and this is b 0; so, 0 1 0 0 1 1 1 0 0 1 0 1 1 1 0 and 1 0 1 ok. So, now the thing is that so for this 0 0 0. So, for this particular case if I consider, so at the time for 0 0 this what is that 0 plus 0. So, this is 0 and for 0 0 1 that is 0 plus 0 plus 1 this is 1 or 0 1 plus 0 that is 1 for this particular case there is 0 plus 1 plus 1 that is equals to 2 then again for this is minus 2 plus 0 plus 0 that is equals to minus 2 then again for these particular case that is minus 2 plus 0 plus 1 that is equals to minus 1 again for this minus 2 plus 0 plus 1 that is equals to minus 1 then again for this minus 2 plus 1 plus 1 that is equals to 0 ok. So, this is the operation which I need if I do this corresponding decoding.

Now, we have taken one example and at that time we have seen at the very last, suppose this now I need to add a into b correct.

(Refer Slide Time: 04:20)

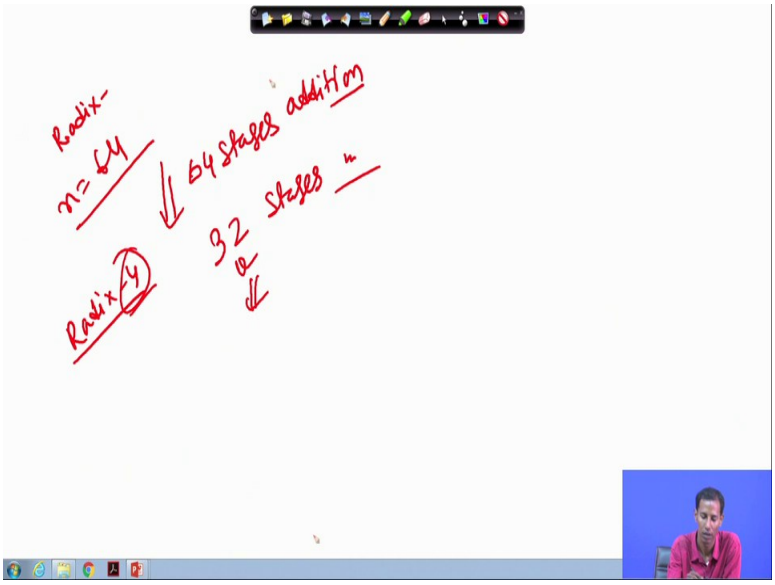



So, b if I consider, so, at the time very end of b I have to add one 0 and then I have to consider the corresponding 3 bit. Suppose if b b is known as let us consider that is b 3 b 2 b 1 b 0 four b, I have to consider another 0 over here, then at the very first I will consider this 3 combination and then there will be one overlap of this ok.

So, whenever I am having initially I am having what? How many times? I am having 4 partial products set ok, 4 partial products set; that means, I need 4 stage of addition sorry addition operation. But in this if I use Booth's recoding that is 4 Booth's recoding, so, at the time PP 1 for this particular case and PP 2 for this particular case.

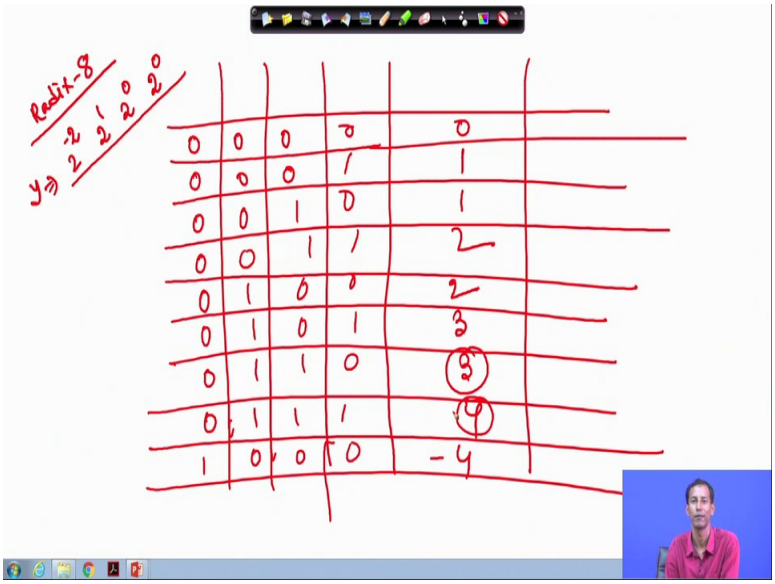
So, at that time I need 2 partial products sets. So that means, here I need only 2 stage of addition operation. So, there means 2 addition operation I have reduced by using this decoding technique ok. So, this is the beauty of this Booth's multiplication, how we can reduce the delay by 50 percent.

(Refer Slide Time: 05:59)




Now, in case of what I; that means, suppose I for if the n is 64, so, at the time what I need? I need 64 stages of addition right? If I use Radix-4 still I need 32 stages of addition operation still it is more. So, at the time can I increase this Radix, so that I can reduce this term? So, if I want to increase the Radix, so at that time what will be its decoding table? So, if I just if you remember based on what based on one particular equation I have; that means, come to what will be the operation.

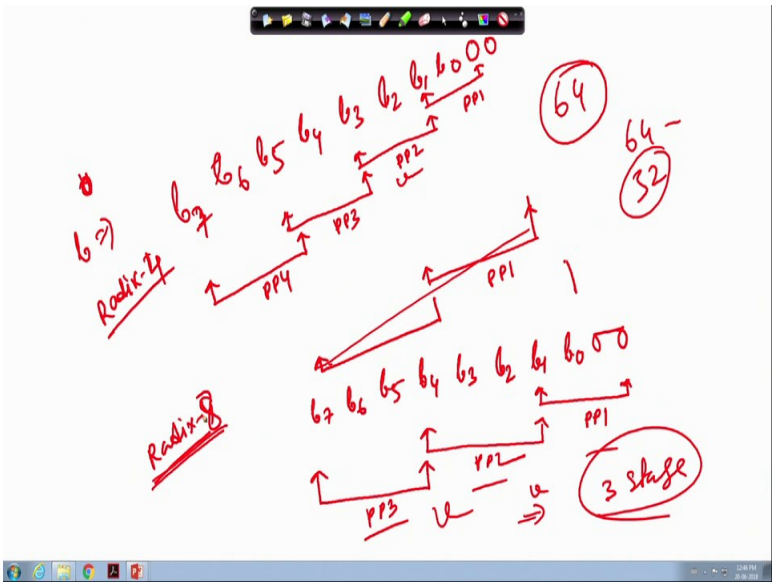
(Refer Slide Time: 06:58)



So, here also if I want to change the corresponding radix, so suppose Radix-8, so at that time what will happen? It will be this at the time y will be what? This will be the position and here the corresponding value will be something like this ok. So that means, here in the earlier case what was there? That means, that was in this particular position that is that is minus 1, here 2 to the power 2 will be the minus case ok. So that means, now considering these things; so there will be 0 0 0 0, 0 0 01, 0 0 1 0, 0 0 1 1, 0 0 1 0, 0 0 1 0 1 something like this.

So, for this case it will be 0 for this case 1 for this case again 1, for this case this is 2, for this case this is 2, for this case this is 3, for 0 1 1 0 again this is 3; then for 0 1 1 1 this is total 4, then again for 1 0 0 0 this is minus 4; so something like this I will get. So that means, now this partial products I have to generate from the beginning ok.

(Refer Slide Time: 09:00)




So, now again whenever we will consider this sorry this b at the time what will happen? Suppose if I consider; that means, b 8 bit ok, b 5, b 4, b 3, b 2, b 1, b 0. So, initially I have to put 0 over here ok. So, if I consider Radix-3 at the time what will be the sets? 1 then 2, then 3, then 4; so that means, this is PP 1, this is PP 2, this is PP 3, this is PP 4.

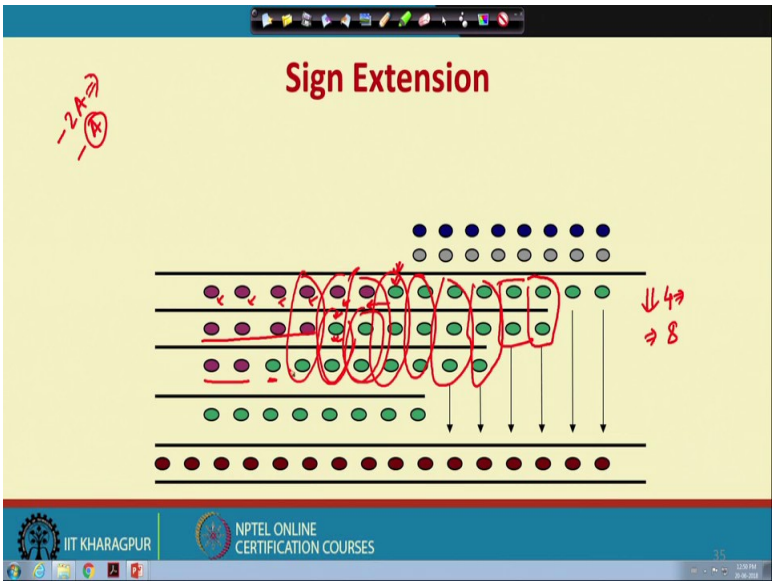
But in case of this is case of Radix-2. In case of Radix-4 what will happen? I have to consider 4 bit. So, at that time what will happen? This is PP 1 and again this will be starting from this not actually not this at the time I have add two 0 over here. So, this will be something like this means what? If I just rewrite, so b 7 b 6 b 5 b 4 b 3 b 2 b 1 b 0

along with these two 0 at that time, it will be something like this. This is PP 1, then this is PP 2 then ok. So, something like this I can do. So, more the number of bits we consider; that means, if we consider 64 bit multiplication or 32 bit multiplication. So, at that time I need to use this is for sorry this is for Radix-4 this is for Radix-8 ok.

So, more the number of Radix I can choose for Booth's decoding. But the thing is that whenever in the previous case, so I will get 16 combination and I need to generate this partial products from the beginning based on the corresponding set. So that means, more the number here I consider, more the number of partial products I need to generate or this decoder size will be also more ok. So, we have to be; that means, choice this number of Radix in a judicious way, so that I am getting the benefit in terms of this delay; that means, stage reduction whenever we are considering or hiring the Radix using this Booth's algorithm ok. So, that we have to be more cautious about.

So, here initially I am need 3, but here I need sorry here I was needing 4 for this 8 bit operation, but here I need 3 stage; that means, only 1 stage I am just reducing. Apart from that, what? But I am increasing the complexity of the Booth's decoding algorithm as I have used Radix-8 over here. So, that thing we have to be careful that whether we are increasing or it is not meant that all the time, if I increase the Radix it will give me the advantage ok. So, that should be whenever you are designing, so at that time you should be more careful that for what purposes for what number of bits you are considering. So, at that time whenever you are considering those numbers of multiplication; that means, the bits for those multiplications. So, at that time what Radix if you choose Radix 8 at the time does it giving you the benefit or it is not. If it is giving you the benefit, so at that time you can go head otherwise you have to be satisfied with the lower Radix ok. So, this is the main fundamentals of Booth's decoding technique ok.

(Refer Slide Time: 13:45)



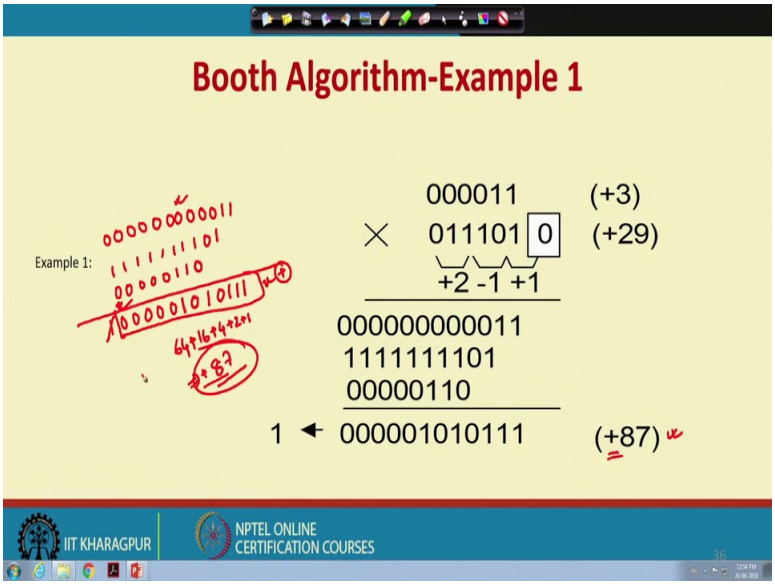

Now, again just here whenever we are doing this Booth's decoding, so at the time what I said? If I choose Radix-8, so at the time the shifting; that means, of this is the first partial product generator. then the next partial product generator that will be added by shifting of 2 bit as we have chosen the Radix of 4. So that means, the next again that is 2 bit.

In generic multiplier, what we do? We shift only 1 bit, but here 2 bit. If we choose Radix-4 8, so at that time 3 bits shift will be there ok. But what about this particular; that means, positions ok? So, this particular position if I filled all this with 0 at that time for the signed value the corresponding results which I will get that will be total erroneous results I will get. So, that is why I need the sign extension of them, why I need this sign extension? Because in the decoding table what I find that sometimes I need 2 of minus A minus 2 of a or minus of A means what? This is nothing but the 2's compliment.

So, minus of a means 2's compliment means this sign; that means, the MSB that has to be extended up to the; that means, maximum MSB sign. That means, whatever the bit position is this, so that will be copied multiple times for this particular position. So, this is for the next set whatever is the bit position over here or MSB of this that will be copied for this, whatever is this the bit of this that will be copied to this 2 bit position and then you do the Wallace tree multiplication; that means, what? Wallace tree structure; that means, then Wallace tree structure means you just add something like this or using carry save adder or using carry look adder error sorry. So, using any of these now you

can just do this addition operation ok. So that means what? If I use generic carry save; that means, Wallace tree multiplier at that time this depth I need is more than here I need 4, but if I use only Wallace tree, so at that time I need 8 stages ok. So, this is the sign extension method which I need to follow whenever we are using these Booth's multiplier architecture ok.

(Refer Slide Time: 16:43)




So, if we take one example using; that means, how we perform this sign extension whenever we are using this Booth's algorithm. Suppose this is the MUL; that means, the multiplicand and this is the multiplier ok. So, this value is plus 3 and these value is plus 29, so the results I which will be plus 87 right. So, what I need to do? I need to add 0. So, then for 0 1 0, so, the case will be plus 1 or 0 1 0 that is plus 1.

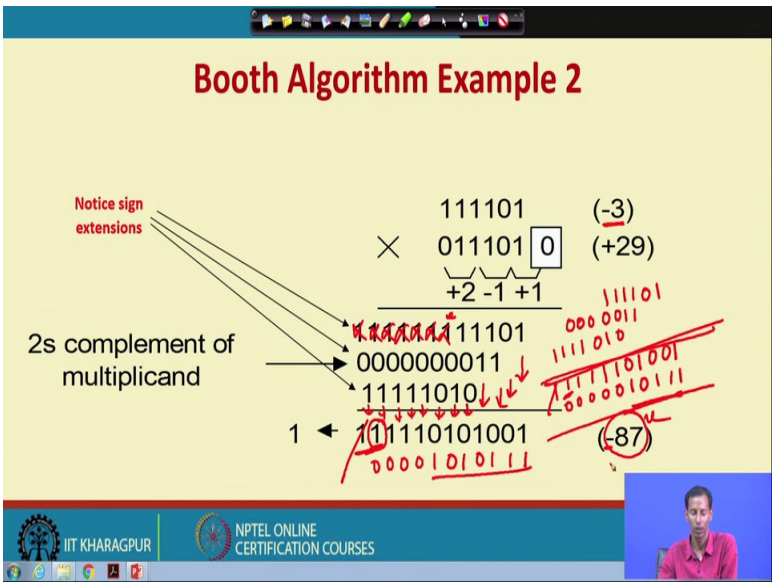
For 1 1 0 the case is minus 1, then again for 0 1 1 there you; that means, the corresponding decoding table says that the multiplier multiplicand will be plus 2. So that means, now for positive of A, then that I have to write, for positive of A I have to write something like this 1 2 3 4 then 1 and for as this is 6 to 6 multiplication, so; that means, total 8 bit will be there. So that means, the rest of the 6 bit position, all will be filled with 0 as this bit is 0 over here.

For negative of this will be what? 2's compliment of this number, so, in and that will be 2 bits shifted. So, 2's compliment of this number is 1 then there is 0 then all 1, but as this bit is 1 so; that means, all the 1 will be shifted to the corresponding MSB side then this is

for plus 2. So, it will be started from this particular case and plus 2 means that is; that means, 1 0 has been added at the LSB position ok, basically plus 2 means multiplying with 2 A means 0 if I add 0, so that becomes 2 A ok. So, 0 that is added to this and then again that MSB is shifted to the MSB side, now I need to add. So, this is 1, this is 1, this is 1, then this position is 0 this position is 1; this position is for 1 1, this is 0; for 1 1, this is 1; for 1 this is 0, for this is 0, this is 0, this is 0, this is 0, this is 1 ok.

So, this carry will be discarded. So, as this all this position; that means, the MSB of this is 0, so this is 1 positive number ok. And what is the magnitude of this? The magnitude of this is 1 plus 2 plus 4 plus 16 plus 64 that is equals to 87 ok. So, the result is 87 plus so, plus 87 is the results that I am getting using Booth's multiplier or Booth's algorithm ok. Now then you change; that means, next if we change the number; that means, plus 3 to minus 29 or any of these number to minus. So, at that time what will happen? So, at that time what will happen?

(Refer Slide Time: 20:41)




Here you see I have changed the number to these 2 minus 3. So, minus 3 means 2's compliment of that. So, as this is 0, this bit is remaining unchanged; so that means, that corresponding decoding will remain same. So, decoding will remain same means for minus 3 as this bit is 1, so that has been extended up to this. So, in the previous case as this bit was 0, so 0 was extended or as in this case 1 is there. So, 1 is extended and for minus 1, 2's compliment of this that will be this so; that means, as 0 s 0 will be extended

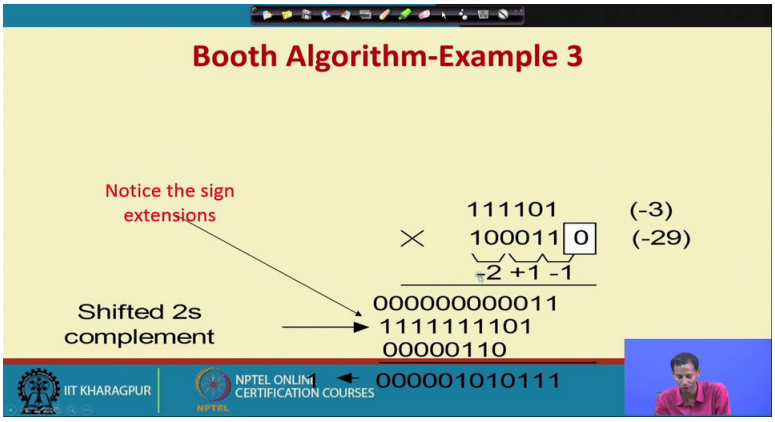
as this particular position. So, again for this case plus 2 means, so 0 is padded and apart from that all are the 1 is extended because this is minus 1 now.

If I just add so at that time what will happen? 1 for this 0 for this for 1 1 0 and 1 carry 1 comes over here. So, this is for 1 then 1 over here for this is for 0 then 1 1 1, this will be 1 then 1 1 there 0; so, 1 1 1 1 1 1 1 1 all 1. So, this 1 means if I just discard the prev; that means, the outer carry. So, the MSB is 1 means this is basically gets this is the negative number which remains in 2's compliment format ok. So, this is negative and if I take the 2's compliment of this then this will become 1 0 1 0 1 all 0. So, this indicates this is the sign which is negative and magnitude is this which is nothing, but the 87 ok. So that means, using the sign extension I am getting the corresponding results of this.

Now, suppose if I don't use this sign extension at the time what will happen? If I don't consider these particular bits, so at that time what will happen? Can I get this minus 87? If I check that? So, at that time if I don't consider this, so at the time what will happen? 0 1 for this case this is 1 1 0 0 and for this case what is the; that means, value is 0 1 0 1 1 1 1 sorry 1 1 again this is all 0 ok.

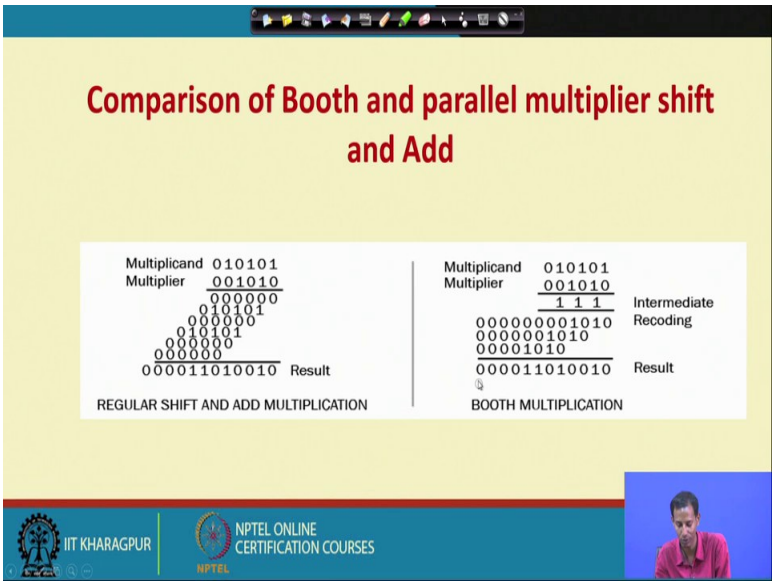
So, now, if I just want to add 1 then 0 this is 0 this is 1 then this is again this is 0 this is 1 this is 1 and then all 1. So, as this bit is 1 this indicates negative very fine, but what about this magnitude of this? So, this is 1 1 1 0 1 then all 0 I am getting. So, is it the number 87? No. That means, I am not getting the corresponding values of this 87 the magnitude is totally different from this; that means, I am getting the results erroneous results; if I don't extend this sign at the MSB position ok.

(Refer Slide Time: 24:53)



So, then again if I consider another example of this where I am both the numbers are different; that means, negative. So, at that time at the results what I need I will get positive value right. So, again here you see this will change. So, at that time so negative of that means now, the corresponding this 2 position will be?

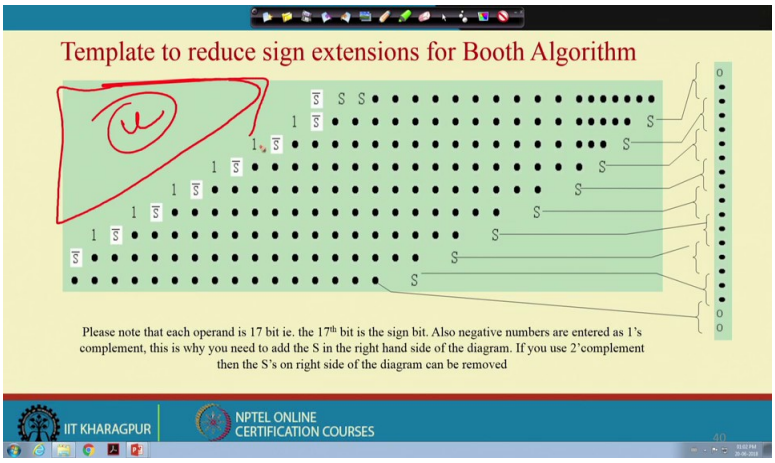
(Refer Slide Time: 25:14)




So, initially this was 1 this was 2 this was minus 1 now this 1 will be minus plus 1 this 2 will be minus 1 sorry minus 1 this will be minus 3. So, now based on that if you just do you will get plus 87, because you are getting 0 over here ok. So, this is the sign extension.

Then what is the problem here? Is there any problem? Yes; obviously, there is the problem. As I am increasing the sign bit for this particular partial products; that means, I am increasing the full adder cell that whenever I will use this to add this particular bits using Wallace tree or any other addition operation, so at the time I am using the computation. As I am increasing this I am putting this value extra bits, so that means, I am putting I am increasing the complexity of the circuit ok. So, is there any way out to reduce that? Can I reduce that? So, that this is regular set and basically this is regular set and this is this Booth's multiplication.

(Refer Slide Time: 26:41)




Can I do that? Yes I can do that. I can reduce this sign extension for Booth's algorithm using this template. So, what type of template? We will see that. Suppose this is the first set and then this is that means, the partial products such something lays generated something like this. So, then it says that please note that each operand is this is considering 8 cross 8; so, 16 bit. If 17 bit that is the 17 bit is the sign bit. Also negative numbers are entered as 1's compliment, this why you need to add the S in the right hand side of the diagram. If you use 2's compliment then the S on the right hand side of the diagram can be removed ok.

So, this is the; that means, instead of putting this S up to this position I can write something like this, so that I can remove the extra burden of extending the sign up to this particular position. So, how we can do or what is the logic behind to putting the; that means, this 1 and S bar? What is the logic behind that? We will take the example and then we will see how we can do that or what is the it is not that from the air it is coming, there is logic behind it ok; so, that we will see in the next class. We will discuss on that how we can use this sign extension. So, means what? Actually what I need to do? That means I need to reduce this extra overhead which are required for if I put if I have not used this kind of notation I have to extend the sign; that means, that means, for this particular position which will be filled with redundant number. So, to remove those I need this techniques which will be I will be discussing in the next class so.

Thank you for today.