Digital Electronic Circuits
Prof. Goutam Saha
Department of E & EC Engineering
Indian Institute of Technology, Kharagpur

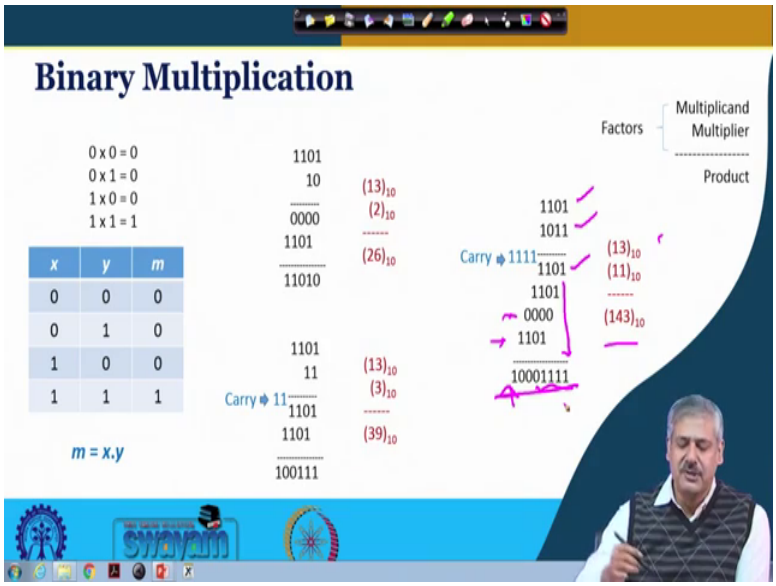
Lecture – 30
Multiplication and Division

Hello everybody, today we discuss Multiplication and Division. You may have wondered we were discussing addition and subtraction and we have not touched multiplication and division. One reason is it is very complex and you need actually repeated number of addition and subtraction. And, we were discussing combinatorial circuit.

(Refer Slide Time: 00:42)

So, there are other ways other approaches use using you know sequential logic circuit concept to arrive at this two. But let us venture into combinatorial circuit based binary multiplication approach and similarly for binary division in this particular class. So, in the beginning we shall look at the concept of binary multiplication, and division and then we shall look at the a 1 such circuit logic circuit ok.

(Refer Slide Time: 01:18)



So, binary multiplication if you understand, this if you are multiplying similar to you know decimal multiplication here only we are using binary digit ok. So, if you are multiplying 0 is 0. So, the result will be 0 of course, 0 with 1 it will be 0 1 with 0 it will be 0 and 1 with 1 it will be 1. So, if we are looking at multiplicand x and multiplayer y which are called both are called factor. And the result is the product then if we put them in the form of a truth table the relationship is x and y is your m ok.

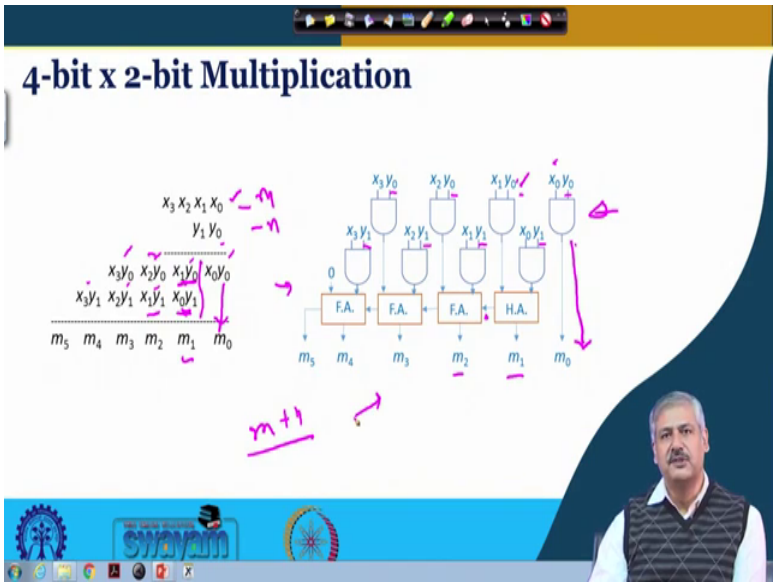
Now this is for 1 bit right, but if you are increasing the number of bits so here is an example of 4 bit multiplicand multiplied with 2 bit multiplier. So, the example is taken is of 13 and 2. So, this 26 you know all right in from decimal multiplication. So, here how do? You do first you multiply with 0 the way you would have done in the case of decimal multiplication. So, 0 multiplied with 1 1 0 1 this will be just 1 bit kind of thing each of this is 1 bit operation.

So, this is the and operation that you do so it is all 0 right. Then we shift it I mean when. So, this we do not consider this particular place and the next position we have 1 multiplied with 1 1 0 1. So, this is 1 1 0 1 only. So, this is what we are having now. So, 0 directly comes here as the m 0 the value in the first place units place and 0 1 1 0 0 0 1 and 0 1 and this is 1 if there is a carry there will be a determine this normal addition is it fine. And we can see this number to be what this is 16, this is 8, 24, and this is 26 it works is not it.

And if we want to look at example we carry when we are doing the multiplication. So, basically this is 13 with 3 it is 39 right. So, 13 13 remains this is 1 1 3 we are multiplying. So, this is your 1 1 0 and this is your 1 1 0 1. So, this 1 directly comes here and rest you have to do by going to the addition process. So, 0 1 1 1 0 1 1 1 0 a carry is generated here. So, this carry and this one so 0 and a carry is generated here. So, this carry is the final carry and you can see the 2 this to be 32 and these 3 together is 7 we know so 39 is it fine.

Similarly, for 4 bit m out 4 bit into 4 bit multiplication if you looking at then we have got an example 1 1 0 1 with 1 0 1 1 so 13 into 11 so this is 143 in decimal. So, here first with multiplication with 1 1 1 0 1 comes then 1 1 0 1 it is just shifted by 1 bit so this 1 will directly come here. Then 0 is multiplied so all 0 then finally, 1 1 0 1. Then you do the adding and if you perform that you will see that it is coming this way this is 128 and these 4 ones are 15 so 143 is it ok. We have understood how binary multiplication is done in reference to decimal multiplication that we are already familiar with.

(Refer Slide Time: 05:05)



Now, if we look at the hardware or the logic circuit for realization of this multiplication. So, first you look at 4 bit into 2 bit ok. So, in this case this is a 4 bit number now it is generalized earlier we have taken specific example so 4 bit number could be anything. So, which we represent using x 3 x 2 x 1 and x naught and this is as y 1 and y naught. So,

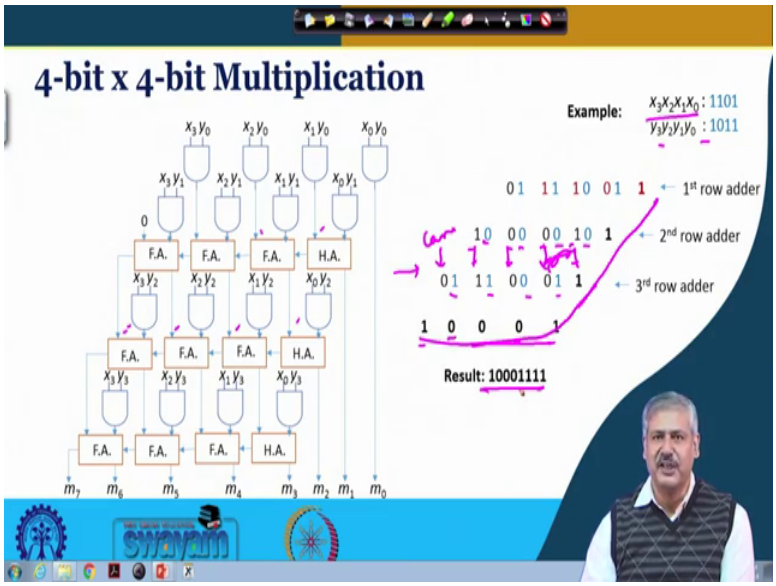
first we are multiplying y x 3 to x naught using y not. So, that is end of this here x 3 with x 3 x 2 x x 1 x naught with y naught bit wise end right.

So, this is represented here this is represented here is it clear ok. And then this y 1 ended with x 3 to x naught so this is there right. And when you perform the addition it this 2 will be part added and this will directly go as m naught ok. And then this addition will come as a 1 and there may be a carry and all those things will be happening. So, if you look at the corresponding circuit 1 possible circuit right. So, first bank of and gate to input and gate right one input of it is y naught in every case you see y naught right.

And the other input is x 3 in the first case 2 x naught in the last case right. So, this is the end output so these end output directly goes these as m naught ok. And whatever is generated these x 1 y naught x 1 y naught over here is to be ended with is to be added with next it is called partial product partial product ok. So, in this case this next case you see all of them are y 1 this is y 1 right and this is x 3 to x naught. So, whatever is generated by this end right x naught y 1 need to be added with x 1 y naught that is getting added right.

And with that whatever output is coming I mean that is going here as m 1 and the carry will be fed to the next place this x 2 y naught x 1 y 1 addition. So, x 2 y naught x 1 y 1 addition so this is coming as carry from the previous stage and this output will be m 2 and so on and so forth this is the way it will go on. So, 4 it number of you know this bits over here is m and this is n. So, totally in plus n number of bits are possible in the product ok. So, this we understand ok.

(Refer Slide Time: 08:00)



So, now we look at little bit more complex thing. So, this is 4 bit L cross 4 bit multiplication ok. So, that is why you see a little bit you know the complexity of it for which we have deserved it towards the end of the combinatorial logic circuit related discussion. so this is the way it has been arranged so stage by stage. So, we have if what if it was 4 bit by 2 bit this is the bank of adder while we had ended the result.

Now 2 more bits are there 3rd bit and 4th bit for which you see another two such bank of adders are there ok. And this will give you finally, m naught to m 7 right. And to understand it let us look at one example since this place is already clumsy I have taken it to the right hand side ok, but it will be difficult to understand and which example shall we take. So, we shall take that 13 multiplied by 11 example ok. And we shall see that how it works in hardware all right in this particular adder based arrangement. So, so 13 x 3 to x naught and 11 is here y 3 to y naught over here ok.

So, the first row this x 3 to x naught so this is your 0 this is the first row adder input that we are talking about so this is the first row adder input. So, this 0 is there in this particular place right and after that x 3 y 1 so x 3 y 1. So, when we multiplied with all this thing it is 1 1 0 1 right. So, this is 1 1 0 1 that you get and the other 1 is other input of the your this particular adder is x 3 and x 2 x 1 x naught ended with y naught y naught is 1.

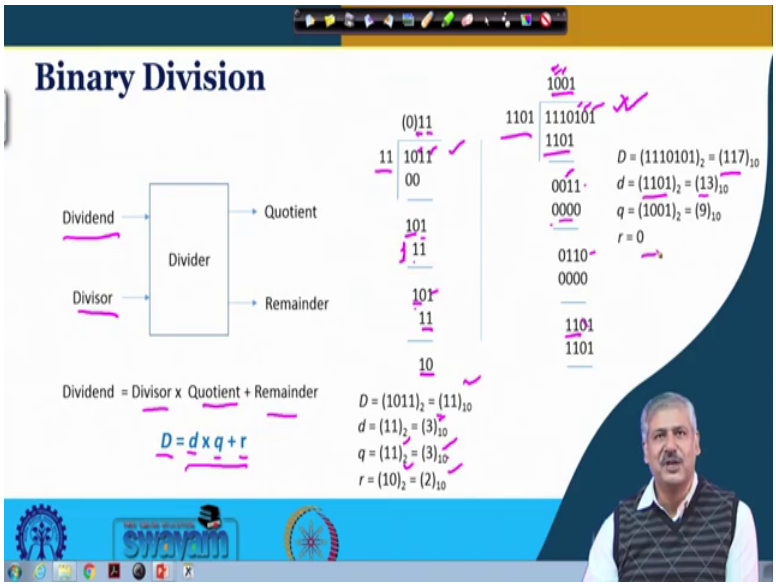
So, that is also giving you 1 1 0 1 right. So, this 1 1 0 1 is a shifted version you see here it is 0 right and here it is not getting added with anyone so this is your 1 1 0 1 right. So, this 1 is finally, directly going here is a output right for the product generation. So, these are the inputs of the this half adder bank of half adder that your bank of adder you are having the first 1 is half adder rest of full adder is it ok. So, when you add it up when you add it up then what you see this is 0 1 is 1 1 0 is 1.

So, this input of the adder ok. So, this half adder this full adder input is 1 of the input of the next adders next set of adders next set of adders. So, this is the input to the next set of adder that is going as the add adder output ok. So, this is what you see in the next group. So, these are the adder output and output from the previous stage this is the carry that you get from the previous stage and then again you multiply it with what y 2 with x 3 x 2 x 1 x naught y 2 is 0 so it will be all 0.

So, that is what we see here 0 0 0 0 right is it fine then again you add it up. And that addition result will come this addition result is coming here this addition result is coming here this addition result is coming here this addition result is coming here. If there is any carry it will it is coming the carry is coming here right. And then again you and it they multiply it with 1 with this 1 one 0 one. So, this is 1 1 0 1 then again you see the result. So, 0 1 this is 1 0 0 1 1 0 that 1 is coming over here carry 1 1 0 and this is 1 right.

So, you collect all these 1 and that is giving you your final result that is 1 triple 0 1 1 1 1 ok. So, due to paucity of time I mean otherwise you can put them over here individual places and see how it is happening right. So, you have just placed side by side, but you can in a bigger picture you can put all those values and keep adding it 1 after another. And you can find that you are able to arrive at this result ok. And for other set of numbers also fine so you can move to division right.

(Refer Slide Time: 13:22)



So, binary division in this case we are having two outputs ok; one is quotient another is reminder. And the dividend is getting divided by the divisor all right. And if dividend is represented by capital D divisor by small d quotient by q and remainder by r then this is what you get ok. And how division can be implemented, implemented it is similar to what you had done in your decimal division. So, if you take an example like say 1 0 1 1 is getting divided by 1 1 ok.

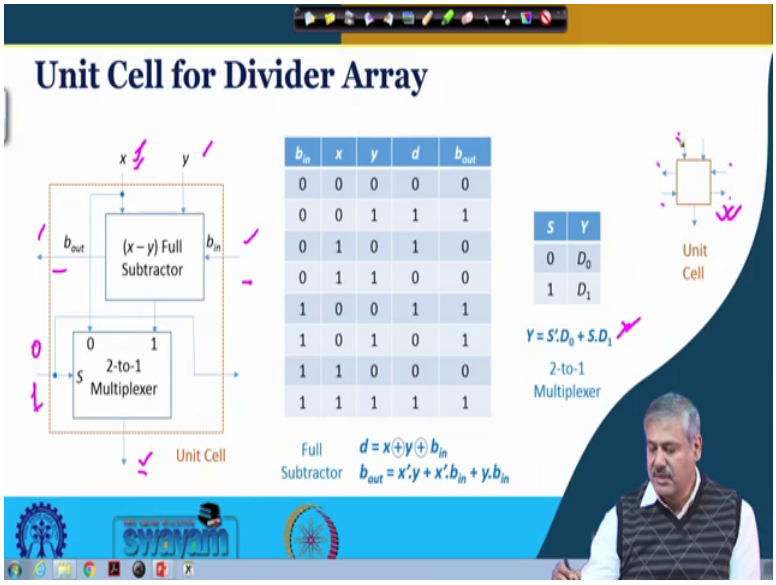
So, first of all this 1 0 is small all right. So, when if you multiply if you have to multiply it with a 0 or the leading 0 does not make any sense. So, this is 0 0 if you subtract you get 1 0 only and then you take the next bit so that is 1 0 1 So, now you can you have 1 over here possibility is only 1 or 0 is not it so 1 means this is 1 1 ok. So, if you subtract this is 0 this is 1 and this borrow that you have taken so borrow was here so 1 1 it now cancels ok. So, you get 1 0 then you take the next value. So, basically this is 1 so this is 1 0 1. So, again a 1 goes there so 1 1 right. And then again if you subtract you get 1 0 and stops here.

So, your quotient is 1 1 all right quotient is 1 1 and remainder is 1 0 right so 1 0 1 1 is 11. So, what you have got is 3 and it was you are dividing with 3 you got quotient 3 and reminder 2 it is fine 3 into 3 plus 2 that is 11 is it ok. So, that was 4 bit getting divided by 2 bit. So, if you have got a example with bigger. So, this is 7 bit getting divided by 4 bit ok. So, the first case you take 1 so basically; 1 1 0 1 here so if you subtract you get 1

right; only 1 and 2 leading 0's right. So, you bring 1 1 from here, but then you this number is smaller right. So, 0 you have to take as quotient.

So, this 0 so it is if you subtract you get only 0 1 1 only. So, you bring next 0 so you still see that the number is smaller. So, you have another 0 over here for two 0 right. So, then subtract 1 1 0 then you bring 1 from here and this is 1 1. So, this is a case where there is no remainder. So, therefore, reminder depending on the value that is there ok. So, the original number 1 1 1 0 1 0 1 this is 170 and you are dividing with 1 1 is 0 1 that is 13 and what you have got here is 9 quotients is 9 and remainder is 0 ok.

(Refer Slide Time: 16:46)



Now to implement it we shall look at again array based structure right. And in this array we shall use the at the cross point a unit cell. So, this unit cell here looks something like this ok. So, we shall explain it what it is right and we shall see 1 example of course, the oh we had seen before. And so for that so this unit cell has got what you see here a input x and y and this is a full subtractor x minus y full subtraction that is there ok.

Subtraction we have learnt using twos complement right, but we can have standard subtractor full subtractor circuit as well ok. So, for example, this is a full subtractor truth table ok. So, this is borrow in this is borrow out this is the difference and this is x from which y is subtracted ok. So, 0 0 this is 0 0 1 x minus y you are talking about so 0 1. So, difference is 1 and borrow is 1 1 0 this is 1; no borrow is required 1 1 this is 0 0. So, this

is 0 0, but there is a borrow from the previous stage borrow in. So, basically it is 0 minus 1 that particular case.

So, this is 1 1. So, this is 1 1 right. So, that is basically 0 minus 2 it becomes 1 plus 1 becomes 2. So, if you would take a borrow from the previous stage so this is borrow right. So, borrow is 1 means 2 it becomes from the previous stage because it is a higher place. So, the difference is 0. So, 1 1 this is the 0 0 and 1 1 1 then basically 1 and you are taking a borrow so which becoming 3 3 minus this is 1 and 1 2. So, basically 1 borrow and 3 minus 2 is 1.

So, this is the way you can get the truth table. And if you look at now the relationship you will see that the difference is coming for the cases where there is only odd number of ones here even number of ones; so, this is 0 ok. So, again here again here so basically this is exhort between x y and v in. And similarly if you look at b out it is x prime y plus x prime b in plus y b in it is similar to carry, but in this carry generation case this was not x prime this was c in carry in ok.

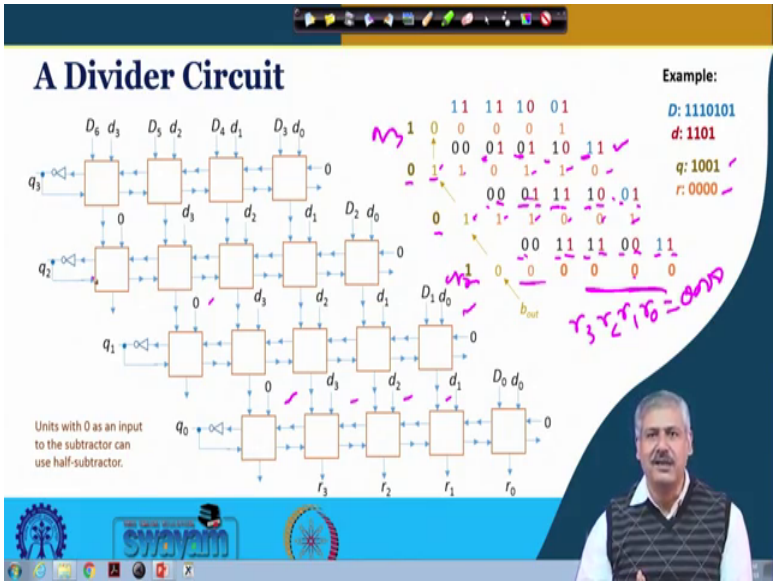
So, this is the way we can get the full subtractor circuit right fine. Now this full subtracted there is input b in and there is output b out ok. And then this out this full subtract output goes to it 2 to 1 multiplexer ok. So, there is a select line 1 select line right 2 2 1 multiplexer circuit and the relationship we already know. So, if this is 0 if the input over here is 0 then x is passed as the unit cell output there is the catch there the nice thing about it the full subtractor algorithm or the approach that we shall see ok.

So, if it is 0 then this one what is to be subtracted in this particular cell that only gets passed not the subtraction result not the not the difference ok. And if this is 1 then the difference gets passed the difference gets passed. So, this is what we need to take note of and this is the thing. And what the other output you see that whatever is being done here same information same select input is going to the next stage. So, there could be other such cell which is asking for since end up in a select input for the multiplexer for that particular stage is it clear.

So, borrow out borrow in the number from which this number to be subtracted and finally, the result which is number from which it is subtracted or the difference ok, depending on the select input right. And this is the way we represent the unit cell right. So, these are the corresponding inputs. So, borrow in select output, select input, or

borrow out, this is number to be subtracted, this is obtained this is manual and this is the output whether it is manual or the difference ok.

(Refer Slide Time: 21:39)



So, again with complex circuit right, but this is doing a division of a 7 bit number by a 4 bit number ok. If you want 8 bit number to be divided by is you know 4 bit number then 1 more stage here over here need to be need to be added ok. So, that is the way it has to be seen ok. And you see that the arrangement is the way we normally divide and divide from the higher bit. So, d 6 is here d 6, d 5, d 5, d 4, d 3, then d 2, d 1, d naught.

So, gradually it is making a right shift ok. So, in sequential logic based circuit development of multiplayer and add you know divider the kind of shift it is required left right these are all taken into consideration ok. so this is the circuit right and now what you can see that this borrow out that was getting generated that borrow out it is inverted form is actually getting fed as the select input of the multiplier ok. So, in the final stage if a carry is generated carry is generated. So, this is 1 so this is then it will become 0 ok. So, when carry is generated then if you go back.

If it is 0 then the number from which it is you know sub subtracted is mine when will come here and otherwise the difference will come is it ok. So, this is what is the case. And this what if part of the carry is what is going to give you the quotient and this is what is going to give you the remainder. So, this is the remainder will come ok. So, this

is the algorithm which has been arrived at and given a given a a shape through a unit cell base array so divider circuit ok.

So, now, we can to appreciate it let us quickly go through an example. So, the example you take is the same 1 that we had taken before that is 117 by divided by 13 all right. So, so this is the dividend and this is the divisor ok. So, dividend then these bits d 6 2 d naught these are there. So, first we shall take up d 6 d 5 d 4 d 4. So, this is 1 1 1 0 so this is d 6 d 5 d 4 and d 3 that is in blue ok. And divisor is 1 1 0 1. So, this is divisor the that is in brown is it ok.

Student: Yeah.

Then what you are doing.

Student: (Refer Time: 25:00).

We are doing the subtraction and in each of the subtraction we are looking at the borrow. So, 0 1 is subtracted from 0 so this is 1 right. So, this borrow is coming over here 1 1 0 right 1 1 0 again 1 1 0. So, final carry is 0 right. So, it is invited is 1 so that is your q 3 ok. So, this is 1 right and this is the borrow out the final that is getting generated that you can see. So, when q 3 is there so the difference result will be going to the next level of subtractor this thing group ok.

So, what that that is what you can see here so this is your 1 1 this is 0 is coming here this the subtraction is 0 the subtraction is 0 all right. In this case the subtraction result was 0 to subtraction it was 1 so 1 is coming here ok. So, these are the bits that you get for the mine went and for the sub trained. So, first place it is 0 so that is the 0 over here and next is your d 3 d 2 d 1 d naught so 1 1 0 1 and for this particular subtractor over here.

So, d 2 gets introduced it is 6 2 d 3 was there so d 2 get is getting introduced here. So, this is in blue. So, d t d 2 is 1 1 0 then it is 1. So, this 1 is coming over here is it fine. So, next you do what? Next you do again subtraction of this particular set right. So, this is your you are here. So, 1 1 0 so 0 to 1 this is 1 0 minus 1 is 1 borrow is 1 1 0 this is 2 right. So, there is a this is 0 so this is again 1. So, this is 0 minus 1 is 1 and then there is a borrow out which is 1 ok. So, this is if there is a barrow out. So, then it is inverse is 0. So, this is your quotient q 2 ok.

So, this inverse is now going to the multiplexer input all this multiplexer input. So, when it is 0 then the mine went not the difference will come to the next level of full subtractor ok. So, here this result 0 will not come so this 0 will come over here ok. So, this 0 will come over here this 1 will come over here and this 1 will come over here not the difference result ok. That is what we have seen the operation of the multiplexer.

So, for now this particular subtractor we are having 0 introduced for as d 1 and what is to be subtracted is again? This 1 1 0 1 in the brown and for this particular place it is 0 only right fine. So, then again we perform the subtraction and see. So, 0 to 1 we have a borrow of 1. So, 1 1 it is 0 1 1 it is 0 1 to 0 this is 1 right. Then 1 to 1 to 0 this is again 1 and then there is a borrower ok. So, again there is a borrow 1 that is present so it is inverse a 0. So, you get q 1 also and then again this is 0.

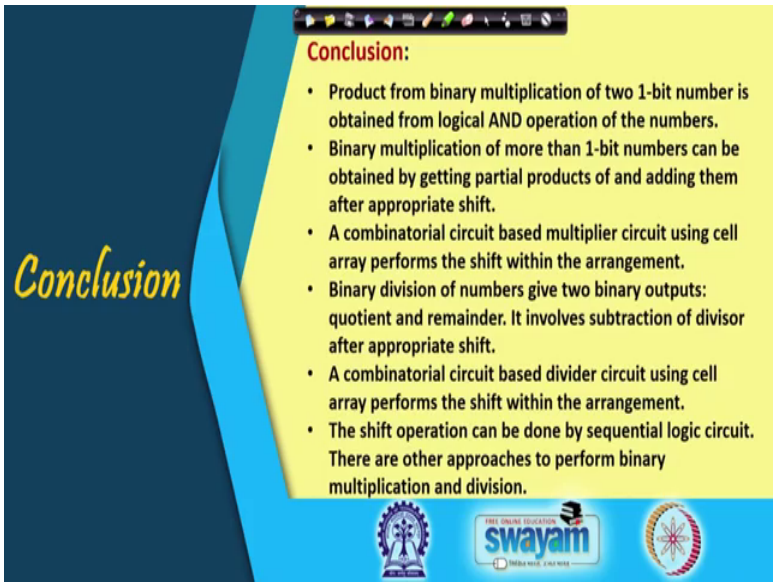
So, what we will see the mine went coming to the next level of subtractors ok, not the difference result. So, that is what is coming over here right 0 this is 1 this is 1 this is 0 and this is 1 is coming as the d naught bit. So, you have got 1 1 0 1 over here is 0. So, to be subtracted this d 3 to d naught so this is 1 1 0 1 ok. Now, if you subtract right you see that this is 0 0 0 all of them are 0 and the borrow is 0. So, it is inverted is 1 and had there been some number present that would have been the remainder.

So, remainder here is 0 all 0 all in I mean r 3 r 2 r 1 and r naught ok. These are 0 0 0 right and your q 3 2 q q 2 q 1 q naught. So, this is your q 3 and this is q naught you take all the borrows right so 1 0 0 1 ok. So, this is the way it is done you can try with some other number also right. And wherever you have seen that only 0 is getting added or the first stage instead of full subtractor. We can have half subtractor right we are not discussed that logic truth table or the circuit, but we can you can similarly get it fine.

(Refer Slide Time: 30:43)

So, with this we come to the conclusion of this particular class.

(Refer Slide Time: 30:45)



So, product from binary multiplication of 2 1 bit number is obtained from logical and operation of the numbers. And for getting more than 1 bit numbers the multiplication result we can get by adding the partial products after appropriate shift. And the combinatorial circuit to get multiplier performs that shift within the arrangement ok. And binary division will generate quotient and remainder and it involves subtraction and of division after again appropriate shift.

So, the combinatorial circuit for that using cell array performs that shift within the arrangement. And as we discussed that; it can be done by sequential logic circuit by some other arrangement which in certain sense can be considered efficient in certain context mode and that we shall take up later so.

Thank you.