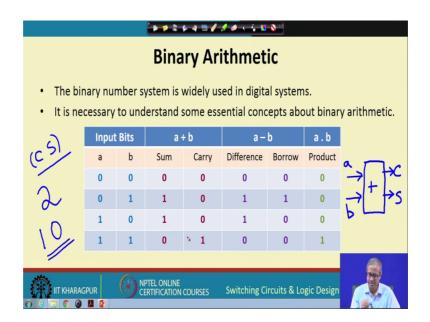**Switching Circuits and Logic Design**
Prof. Indranil Sengupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 04
Binary Addition and Subtraction

If you recall in the last lecture we talked about the different ways of representing negative numbers in binary. So, I mentioned when we talked about the 1's complement and 2's complement representations that one big advantage they offer is that you can do subtraction using addition. You do not need or require a separate subtraction circuit in your system, if you design an adder that is enough you can also do subtraction.

So, in this lecture we shall talk about how addition and subtraction can be carried out in binary and how 1's complement and 2's complement representations can help us in that ok.
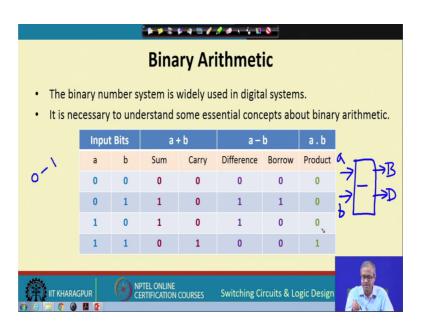
(Refer Slide Time: 01:11)

So, let us look into this generally speaking when you talk about binary arithmetic, let us simply consider 2 bits suppose I have a circuit I am treating it as a black box. So, I am applying 2 bits in the input these 2 bits are a and b. So, in this table this is sometimes the truth table we shall talk about it later. So, I am showing the input bits a and b and I am showing all possible combinations of a and b. There will be four possible combinations 2 to the power 2 0 0 0 1 1 0 and 1 1.

Suppose we are trying to design an addition circuit, we are trying to add them. So, when you add 2 digits my result of addition will also be 2 bits in the output. So, one of them I call it as sum, one of them I call it as carry. So, how does it work see in decimal if you think 0 plus 0 is 0 so, the sum will be 0 no carry, 0 plus 1 is 1. So, my sum is 1, but no carry similarly 1 and 0 if you add result will be 1 no carry, but if we add 1 and 1 in decimal the result is supposed to be 2.

So, in binary what is 2? In binary 2 is represented by 1 0. So, we say that my sum is 0 and there is a carry of 1 ok. So, my output binary number 2 digit number is considered as carry followed by sum, 1 and 0 right. This is how addition is carried out on this a and b.

(Refer Slide Time: 03:22)

Similarly, if I do subtraction now I am saying. Similarly, if I do subtraction now, I am saying that my circuit is such that I am doing subtraction. So, again my 2 numbers are there a and b. So, now, we have the two outputs one I am calling as difference and other I am calling as borrow.

So, again you see if you subtract 0 from 0 the result is obviously, 0. So, difference is 0 borrow is 0, but if you subtract 1 from 0 0 minus 1 which means you are taking a borrow. So, in this case both difference will be 1 and borrow will be 1 this is the rule of subtraction in binary; that when you are subtracting 1 from a 0 you will have to take a borrow from the previous stage.

So, the difference will be 1 and also because you have taken a borrow, borrow will also be 1. But if you subtract 0 from 1 result will be 1 so, difference is 1 borrow is 0. If you subtract 1 from 1 result is 0 so, it is 0. And finally, if you do multiplication let us say if you are doing multiplication of a and b it is fairly simple. So, anything multiplied by 0 is 0. So, first three will be 0 0 0 and 1 multiplied by 1 is 1 ok, but here we are more concerned about addition and subtraction in the present lecture; this product we shall see later.

(Refer Slide Time: 05:06)

Let us talk about binary addition first and let us work out some example to explain the process. So, we are familiar with decimal addition when you add 2 decimal numbers. Let us say when you add let us say 26 and 37 what you do, we add 6 and 7 13, 3 and 1 becomes a carry then we add 3 plus 2 plus 1 6 and no carry this will be our result. So, for binary we do something very similar. The corresponding bits are added and if there is a carry it will be added to the next pair of digits on the left.

So, let us try to do some trail addition. For the time being assumed that the numbers are all positive sign is not there, let us take 2 5 bit number 0 1 0 1 0 and 1 0 1 1 1. So, initially there is carry 0 or 1 are added 1 no carry is 0 0 1 1 means 2 means 0 will be the sum and 1 will be the carry, 1 0 0 again 2 0 with a 1 carry 1 1 0 again 2 0 with a 1 carry 1 1 0 1 0 with the 1 carry.

(Refer Slide Time: 06:38)

So, this will be my final result this is my 5 bit result and there is a carry which is finally, come out right.

Let us take another example. So, in binary this means this number represent 15 plus 1 the result should be 16. So, let us again initially no carry 1 plus 1 0 with a carry of 1 1 1 0 you add 0 with a carry of 1 again 0 carry of 1 again 0 with a carry of 1 1 0 0 is 1 no carry so, this is my result. Let us take another example similar 1 0 is 1 no carry 0 0 1 is 1 no carry 0 1 1 0 1 1 0 is 0 1 0 is 1 this is the result. So, this way you can add any numbers by keeping track of carry's that is generated.

So, the point to notice that at every stage after of course, the first one you are required to add 3 bits, these 2 bits you are this adding plus the carry which is coming ok. This is the process of binary addition simple, just like decimal addition.

Now, talk about subtraction. Subtraction is a little more complex because you need to means understand how the borrow bits are generated. Let us take 2 examples, 3 examples in fact. So, I am subtracting this second number from the first number 0 minus 0 is 0 no borrow, this is the borrow no borrow this 1 minus 0 is 1 no borrow.

Now, 0 minus 1 0 minus 1 if you recall it from table difference will be 1 and also borrow will be 1. So, because borrow is 1 so, effectively you are subtracting 1 from 1. So, it becomes 0 and since we have taken a; borrow this 1 will not be there finally, it will be 0 minus 0 0 because this borrow has been taken out.
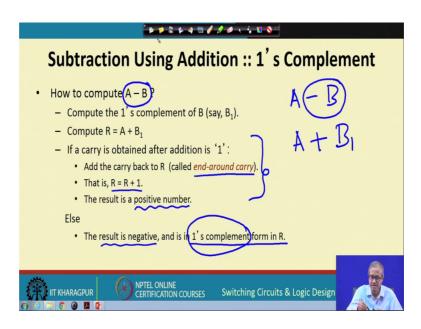
Similarly, there is a another example 1 minus 1 is 0 no borrow 1 minus 0 is 1 no borrow 1 minus 0 1 no borrow 0 minus 1 is 1 there is a borrow, because you have taken a borrow this 1 disappears so 0 minus 0 is 0.

The third example 1 minus 0 1 minus 0 0 minus 1 1 with a borrow you have taken a borrow already here ok. So, again you are subtracting this 0 minus 1 because this is a borrow 0 minus 1 will be 1 again a, borrow again 0 minus 1 1 with a borrow. So, here your sum your result will be this with a, borrow of 1.

So, you see binary subtraction is a little more complicated because the rule for the borrow is not as easy as in case of decimal subtraction ok. So, what you are trying to do is that we try to avoid subtraction all together and say that well if we use 1's complement

and 2's complement, we can do subtraction using addition only. We do not need to remember how subtraction is done ok.

(Refer Slide Time: 10:35)

So, you can see that will be pretty nice this kind of complex rules you do not have remember anymore ok. Let us see subtraction using addition in 1's complement first let us look at this; the rule is as follows suppose I want to compute A minus B, A is binary number B is another binary number I want to subtract B from A.

So, when I want to subtract B from A what I am say telling is that you take the 2's complement of B, 2's compliment you take the 1's compliment of B. Let us call it B 1. So, instead of subtraction you add B 1 with A 1's complement of B let us call it B 1, you add B 1 to A. Then there is a correction step, this actually is illustrating in the example, but after this addition if you see that there is a carry coming out then you add that carry back to the result. This is called end-around carry, but if there is no carry you do not do anything.

So, if a carry is generated you can deduce two things, first is that the result is a positive number and of course, that carry has to be added back to get the correct result. But if there is no carry else part, the first thing is that your result is negative and whatever result you have got it is already in 1's complement for that negative number. So, using addition only you are able to do the subtraction.

Let us illustrate this with the help of an example.

(Refer Slide Time: 12:26)

Let us take a simple example of subtracting 2 from 6, let us consider 4 bit representation; 6 is 0 1 1 0 and what will be the representation of minus 2, 2 is 1 1 0 1 2 is 0 0 1 0. So, if you take 1's complement to 2 flip the bits 1 1 0 1 1 1 0 1. So, you add 1 1 0 1 to this is your B 1. So, as per the previous thing this is your B 1. So, if you add using the rule I have just mentioned 0 plus 1 is 1 no carry is 1 plus 0 is 1 no carry 1 plus 1 is 0 there is a carry 1 plus 1 is 0, there is a carry this carry comes out.

So, now you see there is a carry. So, this is called end-around carry you take this carry back and add this carry with this number 0 0 1 1 plus 1. So, 1 on 1 is 0 with a carry of 1 1 and 1 is 0 with a carry of 1 1 plus 0 is 1 no carry 0. So, the final result is 4 0 1 0 0, 6 minus 2 supposed to be 4 ok. So, we have done subtraction using addition only ok.

Let us another example where there is no end-around carry; that means, a number that means a result is negative.
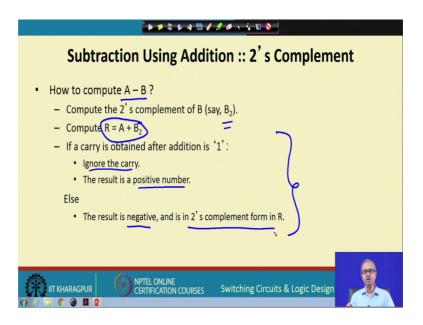
(Refer Slide Time: 14:12)

Let us take 3 minus 5. So, 3 can be represented like this 0 0 1 0 and 5 is 0 1 0 1. So, 1's complement is 1 0 1 0 add them up 1 plus 0 1 and 1 is 0 with a carry of 1 1 0 0 1 no carry 0 1 is 1 no carry.

So, here this no carry you do not do any nothing leave it as it is and you can check from the table 1 1 0 1 is nothing but the 1's complement of 2; that means, minus 2. So, your result is already there. So, so if there is a end-around carry you will have to add it back; if there is no carry you do not add anything your result will already be in 1's complement ok. This is what you mean by you can do subtraction using addition.

(Refer Slide Time: 15:13)

Let us look into the 2's complement representation of same thing. Here the idea is very simple simpler than 1's complement in fact. So, to compute A minus B you compute the 2's complement of B, let us call it B 2 add at B 2 to R. Here there is no concept of end-around carry you have to adding back, nothing it is required. So, if a carry is obtained simply ignore the carry, the result would be positive number.

And if the carry is not there in the result is negative and the result is already in 2's complement form, which means no correction step is required here like in 1's complement. That is why I told you 2's compliment is much more convenient and almost all computer systems today use 2's complement representation for storing negative numbers and also manipulating carry carrying out arithmetic on negative numbers ok. This is the reason.

The second correction step as was required in 1's complement is not required here ok.

(Refer Slide Time: 16:36)

Let us take an example that same example 6 minus 2. So, we have 6 minus 2 so, again plus 2 is 0 0 1 0. So, 1's complement of 2 is 1 1 0 1. So, 2's complement will be you add 1 1 1 1 0 so, this is 2's complement. So, you add this up 0 and 0 no carry 0 1 1 is 0 with a 1 carry 1 1 1 means 1 with a 1 carry 1 0 1 0 with a 1 carry this 1 comes out.

Because a carry is coming out you simply ignore the carry whatever remains that is a result 0 1 0 0 is plus 4. So, you can see this is much simpler than 1's complement; you simply add, ignore the carry whatever is there that is the result right.

Let us look at another example when the result is negative.

Let take 3 minus 5. So, 3 is 0 0 1 1 and 5 again 5 is 0 1 0 1. So, 1's complement is 1 0 1 0 to take 2's compliment add 1 1 0 1 1 this is 1 0 1 1. So, add 1 0 1 1 1 and 1 is 0 with a carry of 1 1 1 1 is 1 with the carry of 1 again 1 0 0 is 1 no carry 0 0 1 is 1 no carry. So, you are left with 1 1 1 0. So, you follow the rule what is 1 1 1 0 represent 2's complement, I mentioned the weight of the MSB will be negative. So, the value will be minus 8 plus 4 plus 2 the last one is 0 so, there is nothing here. So, this is minus 2.

So, you see simply do an addition you do not have to worry about anything your result will automatically come correctly. But the only thing is that all negative numbers you have to represent in 2's complement form that is the rule you to follow. So, if so if in a computer system all your numbers are already stored in 2's complement form you need not have to bother about anything.

So, when you are subtracting you take 2's complement add it otherwise you simply add it numbers can be negative, numbers can be positive there is no issue. This is the big advantage of 2's complement that you do not need subtraction at all ok. So, when we later on when we will be designing circuits we shall be ignoring subtractor circuits all together, we shall only be constructing on designing adder circuits because we know if we have an adder we can also do subtraction.

(Refer Slide Time: 20:03)

So, the last thing we talk about today is that suppose we are doing an addition it may so happen so, let us take a small example. Suppose there are two positive numbers ok, let us say 0 1 1 0 this represents 6 and let us say 0 1 0 1 this represents 5; I want to add them up it is 1 1 0 1. Now, you see 6 and 5 is supposed to be 11 now, if I add them up you see the most significant bit is becoming 1 which indicates the number is negative.

So, why it is becoming negative? Because you see the sum is supposed to be 11. Now, in 4 bit 2's compliment I cannot represent 11, I can represent numbers in the range plus 7 up to minus 8 that is the range.

Now, here the result after addition is coming to coming to 11 which cannot be represented. This is referred as overflow ok.

(Refer Slide Time: 21:26)

Let us take another example, let us take some negative numbers; let us say 1 0 1 0 which represents you can check this is minus 6 and let us say 1 1 0 0 this is minus 4. So, here again if you add 0 1 1 this becomes 0 with a carry of 1 carry or ignoring, you see the numbers were negative but the result this is 0 number the result has become positive.

So, again when we add this the result is supposed to be minus 10 which cannot be represented in 4 bits; this again is overflow ok. So, how do you detect overflow? Detection overflow is not that difficult, you see the overflow cannot occur if one of the number is negative or there is positive and if you add them there can be no overflow.

Over flow can only happen when both the numbers are negative or both the numbers are positive, then only it can go out of range ok. So, the sign of the two numbers must be same this is the first requirement and the sign of the sum is different from sign of either of the numbers which was happening here.

In the first example the sign of the two numbers were 0 0, but the sign of the sum was becoming 1. In the second example numbers were negative they were 1 and 1, but sum the sign was 0 so, they are changing. But there are other ways to check also for the last addition stage, if the carry in and carry out you see they are different that will also indicate there is an overflow. So, there are multiple ways in which you can check but the essential idea is same; the two numbers must be of the same sign and after addition the

result will become of the other sign that indicates that there is an overflow on addition ok.

So, with this we come to the end of this lecture. Now, in the next lecture we shall be talking about some of the other codes that we use in practice typically for representing decimal numbers. There are something called BCD codes, grey codes. So, we shall be looking at those special kind of codes in our next lecture.

Thank you.