Digital Circuits and Systems
Prof. S. Srinivasan
Electronics and Communication Engineering
Indian Institute of Technology Madras
Lecture # 13
Subtractors

In the last couple of lectures we talked about adders as the basic building block of arithmetic circuits which form a major unit of computers. There are other arithmetic operations but adder is only one of the elementary operations. If there are other arithmetic operations except subtractors, subtraction, multiplication so you can have arithmetic circuits designed for each of these operations. The concept is almost the same, what you do in the case of adders is we have to write the truth table and we have to get the Karnaugh Map and get the simplified version and then implement it and if possible try to manipulate the simplified version to a form which is more amenable for practical implementation like Exclusive OR gates.

We also saw methods of speeding up because speeding up is important. In the case of adders we saw there is a technique called Carry Look Ahead and we also mentioned that there are other techniques which we are not going to get into. Likewise you can always do for subtraction and multiplication.

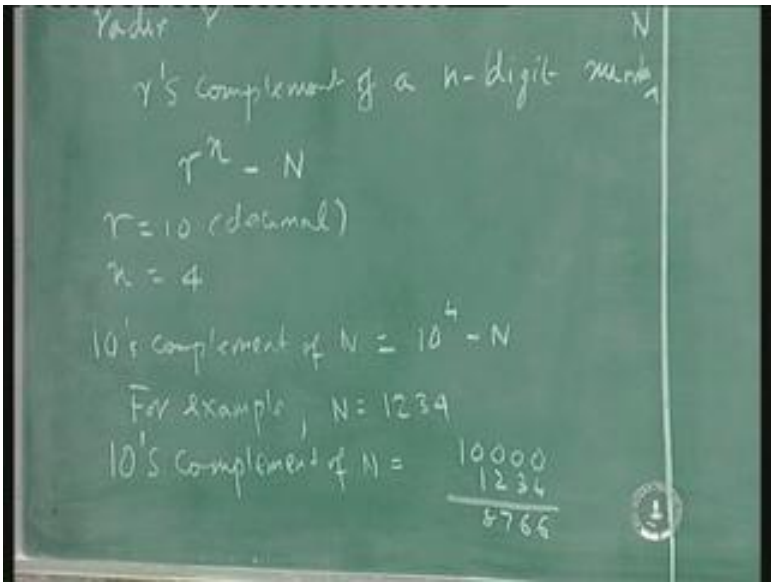
So, if you want to build a subtractor for example you have the half subtractor and full subtractor which are only going to subtract two bits without considering the previous bit, you have a half subtractor and if you want to do a subtraction of series of bits then when you are considering the subtraction of the previous bit you will have to see whether one has already been borrowed from that location to the previous location. So just as you have 'Carry in' with the case of adders we will have a borrow in in the case of a subtractor and similarly we will have a borrow out so we can have like adders two inputs and a 'Carry in' and sum and carry out as outputs, you can also have for subtractors two inputs and a borrow in and the difference between the two bits and borrow out.

You can draw similar truth tables and similar circuit implementation using Karnaugh Maps. We will not get into that because we have already seen large number of Karnaugh Map techniques or truth table techniques, doesn't matter what the function is, once you define a truth table once you define the Karnaugh Map implementation is straight forward so there is no point in going on talking about different types of circuits. Of course more number of circuits you design you get more practice. But there is another scheme of subtraction called complement subtraction. That is because the hardware in your computer you want to have a hardware arithmetic logic unit which does the arithmetic and logic functions you will to have that as compact as possible you don't want to have the adder unit and subtractor unit like that. So if it is possible to combine more than one operation using the hardware unit that is sort of a desirable feature in

hardware design. So, for this purpose now we will talk of what is known as subtraction by complement numbers, subtraction using complement numbers.

What is the complement of a number? It depends on the number system. In any number you have a base, for decimal system the base is ten, binary system the base is two, they also call it based or radix. Therefore depending on the number of digits or the positions or the bits you define the complement as the highest possible number you can represent minus the number.

(Refer Slide Time 09:21)

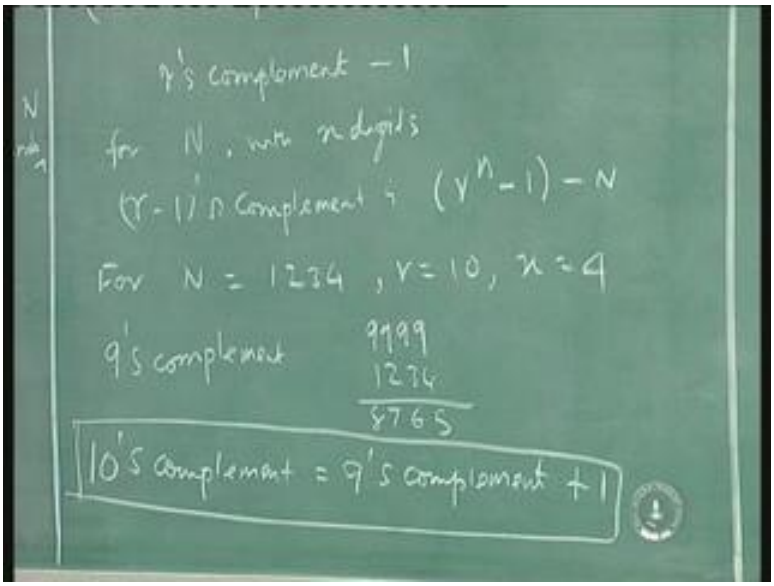


So let us say in the case of any general radix r r's complement of a n bit number, n digit number or n where digit is for decimal, bit for binary so r can be anything I don't know whether to call it a digital bit or I can say n position whatever, the number is r power n minus the number. So r is complemented by a number N n digit number N, I have used in the wrong variable that's ok small n is small variable there is another capital N, N is the number of digits.

What I mean by this is if for example let us say r is equal to 10 decimal system and let us say n is equal to 4 can be four digits so if I represent the number N is 4-bits digits the 10's complement of N which is a four digit decimal number is 10 power 4 minus number N. If the number N is 1 2 3 4 four digits a decimal system then the 10's complement of this is 10 power 4 is equal to 10000 subtract 1234 from this so it is 8766. So you just subtract the number from the number which is represented by the base to the power of the number of digits, the base or the radix to the power of the number of digits is the number from which you subtract the given number then you get the complement of that number in that system. There is also something where you define (r minus 1)'s complement same r radix (r minus 1)'s complement is defined as r's complement minus 1. Or you want to put it the other way it is of n, for N (r minus 1)'s complement is with n bits for the number N with n-digits again I could have probably used p-digits but the choice is n it

doesn't matter small n is the number of digits and capital N is the number. Therefore (r minus 1)'s complement is here in this case you can write it as r power n, since I have to subtract 1 from it r minus n minus 1 and from this I subtract N.

(Refer Slide Time 14:55)



So I will apply the same for the same number N is equal to 1234 and in this case r is equal to 10 and n is equal to 4 again small n is 4 so I will use same n is equal to 4, r is equal to 10, N is equal to 1234 but instead of r's complement or 10's complement I will now define r minus 1 complement which is 9's complement, (r minus 1)'s complement is 9's complement so 9's complement is, this is r power n which is 10 power 4 which is 10000 but I have to subtract 1 from it so it is 9999 so it is 9999 minus 1234 which is now 5678. You look at this number 8766 and 8765 there is a 1 difference which is that r minus n minus 1.

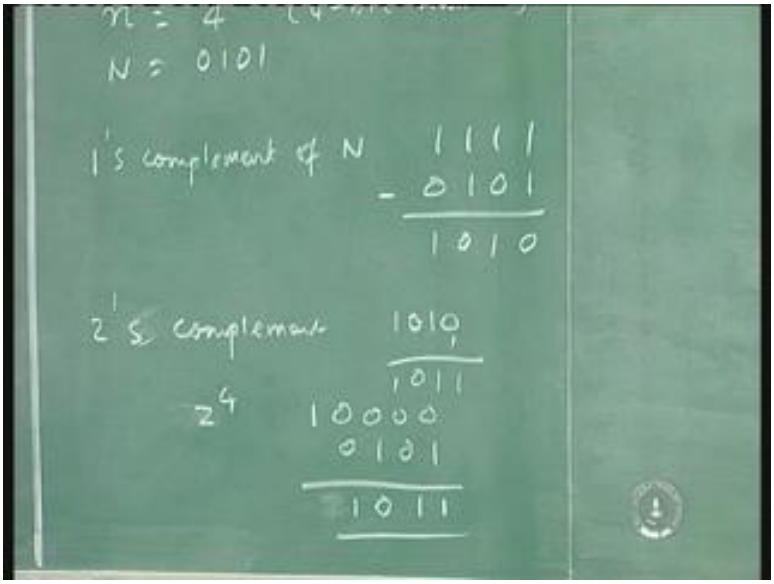
Since I subtracted from r power n minus 1 for (r minus 1)'s complement I subtracted the given number N from r power n minus 1 for r's complement I subtracted the given number from r power n so that extra 1 in my total from which I subtracted this number that 1 is showing off here as 1 less than the r's complement. Hence for a 9's complement of the given number why it is convenient is because 9's complement is nothing but the complement of each digit.

Here I have to make a subtraction of 10 minus 4 so a borrow is involved, 10 minus 4 is equal to 6 so since it is 10 I have to borrow 1 from here so when I borrowed 1 from here this became nine and 9 minus 3 is equal to 6 and since this had to be borrowed so this borrow traveled through, and here there is no such borrow condition because maximum digit I can have here is 9 the maximum value of any of these digits is 9 and this is 9, 9, 9, 9 so I am going to subtract any number so this digit cannot be more than 9 so all I have to do is simple subtraction digit by digit so (r minus 1)'s complement or 9's complement in this case is easier to perform than 10's complement which involves some subtraction in

borrow so if you want to calculate 10's complement I can always finds 9's complement and add 1 to it. Thus 10's complement of any number is 9's complement of that number plus 1.

Before we go to the reason for doing this sometimes subtraction can be done using complement system. There must be some of obvious advantage. We talked about hardware, we talked about arithmetic units which needs to add, subtract, multiply, divide and everything and I want to reduce the hardware or use the common hardware for doing more than one function in that context I am talking about complement numbers which means subtraction using complement numbers can be done using the same hardware that we use for addition. That is why I am bringing this concept, I will tell you how. Before that we have to switch to binary because we will be doing binary arithmetic in our digital codes, logic design or computer design course so this was to explain to you because since you are familiar decimal numbers I introduce this concept of complementing of numbers using decimal concept but in reality we are interested in binary numbers. So binary numbers we have only r is equal to 2 that means I have 2's complement and 1's complement so r's complement (r minus 1)'s complement corresponds to 2's complement 2 minus 1 which is 1's complement so again r power n number of bits minus the given number N in the case of 2's complement r power n minus 1 minus the given number N is 1's complement.

(Refer Slide Time 22:16)



So let us take a simple example without going to all these details and definitions which you have already done in the case of decimal numbers. We will take an example of n is equal to 8 or even 4 make it simple 4 that means a 4-bit number, in decimal you call it digits and in binary you call it bits. I have a number N, can be anything, it has to be a binary number, in decimal I took it as 1234 in binary I need to write it as a binary number so let us say it is 0101 which is 5 really decimal 0101 is decimal 5. So let me find 2's complement and 1's complement for this number so n is equal to 4, N is equal to 0101 so

2’s complement is or I will do it the other way now to make it easy for you, let me first find 1’s complement. What did I say for (r minus 1)’s complement? I write the largest number possible with the given number of digits.

In the case of r’s complement of course I need to write plus 1 it is that base, all these I won’t call it confusion, the difference between r and r minus 1 comes because we talk of 0, the 0 has to be counted if you don’t have 0 then there is nothing like r’s complement (r minus 1)’s complement but 0 is very important we have to represent 0 but you know that is India’s contribution to Mathematics not 0 I think, 0 was contributed, I think now it looks like 0 is the contribution. The 0 is important so now it is 1’s complement.

What is the maximum binary number the largest binary number I can represent using 4-bits 1111. I subtract N which is 0101 from this I get 1’s complement add 1 to it I get 2’s complement. So 1’s complement is 1111 minus 0101 now again my binary arithmetic does not require borrow because there is always 1 at the top and either a 1 or 0 in the bottom so either it is 0 or 1 result so I don’t have to worry about borrow from the next location, next bit position. It sort of makes sense because maximum with 4-bits 1111 is 15 and this is 5 this is 10 and the total should make 15 that’s all, there is no big magic here.

We are giving a number and we are finding its complement, you know what a complement is complementary angles and all that you have studied in trigonometry. So if something is given what is not given is the complement from the total. You subtract what is given from the total you get the complement. The total is 1111 given is 5 so the complement is 10. and 2’s complement is 1010 so we add 2’s complement and the given number should give rise to 1 more than 1111 and 1 more than 1111 is 16 where original 1111 is 15 and 1 more than 1111 is 16 this is 11 given number is 5 so total is 16 that also tallies.

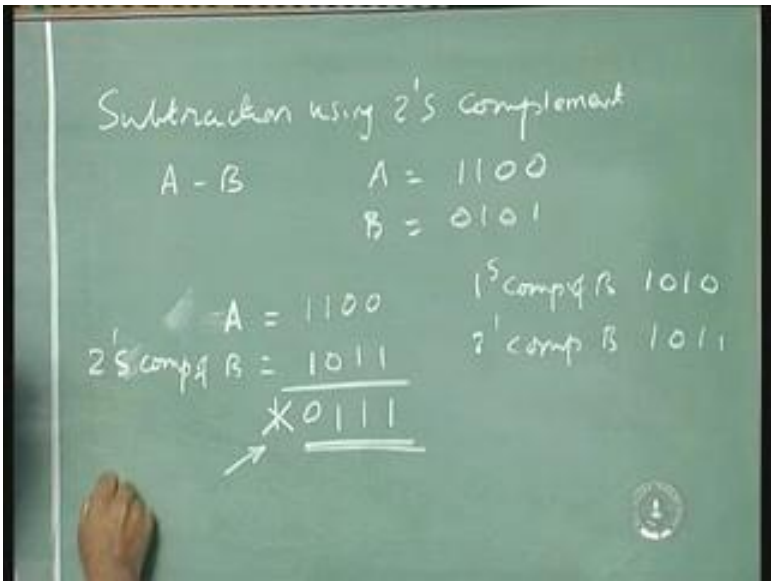
So this addition of 1 gives me 2’s complement that is because now if you want to do it in a rigorous way it is r power n in this case r is equal to 2 where 2 power 4 is 16 so 16 in binary is not 1111 four bits is all right but since 0 is counted 16 becomes the first bit of the next position the fifth position so I need 5 bits to do that which is 10000 subtract 0101 I do the same one so now you have to use the borrow arithmetic and then there is 1, same thing here (Refer Slide Time: 21:46) because it has been borrowed here so this is 1011 16 minus 5 is equal to 11. So I can always find out the 2’s complement of the given number, we need 2’s complement and 1’s complement in our subtraction we will see that. We want to do subtraction not by designing a subtractor.

Subtractor can be designed as we design the adder starting from the requirement, problem statement, truth table, Boolean algebra, Karnaugh Map, reduce and gate structure, it is fine, there is nothing wrong in doing subtraction like that don’t think that it should not be done that way. because you want to have a common hardware for adder and subtractor functionally you want to make it as compact as possible because it is very unlikely that we are going to add and subtract the same time, very very unlikely that in a computer, in a arithmetic unit, in a calculator you are going to do the subtraction and addition

simultaneously you will do one after the other so I can use the same hardware piece. Hence when you do subtraction by complement method I can show that the same adder can be used to do subtraction that is the objective of the whole exercise now.
Therefore given the number you are asked to find the 2's complement or 1's complement no problem you complement the given number bit by bit and then you get 1's complement, add 1 to it to get 2's complement.

Now where does it fit in the case of subtraction? Subtraction is the same as adding the complement of the number. The 2's complement number if you add to a given number suppose I want to subtract B from A and A minus B I want to find out instead of designing a logic for subtraction I can use an adder logic and add instead of B the complement of B, A plus complement of B is same as A minus B. That is why you need to know complement so that we can use the same adder which you have designed whether it is a straight adder or Carry Look Ahead adder or whatever adder it is the same hardware of the functional unit is called adder that can be used for subtraction.

(Refer Slide Time 27:10)

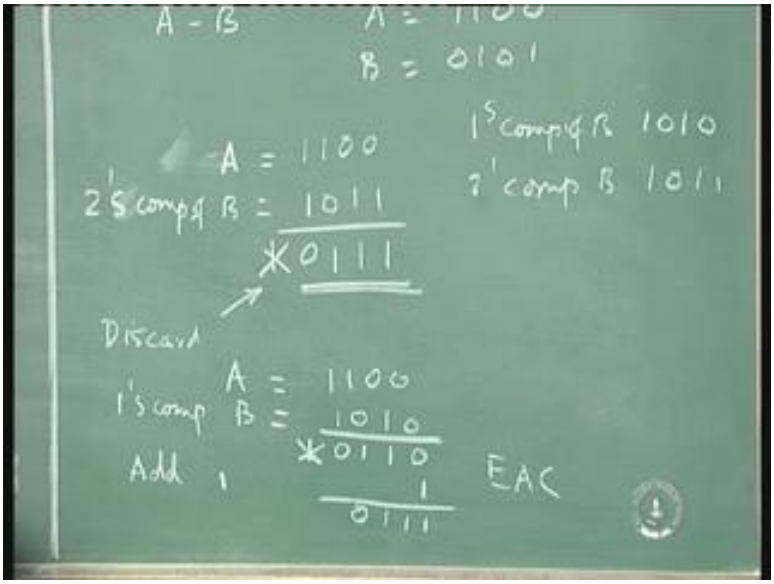


So let us do this example; subtraction using 2's complement so let me again take a simple 4-bit example. Now I want to subtract A minus B let us say A is equal to 1100, A is 12 and B is 5 so 12 minus 5 is equal to 7 I should get as the answer which is 0111. So what I should do is write A as it is I find 2's complement of B so 1's complement of B I first write so 1's complement of B is 1010 that means complement bit by bit, each bit you complement that is 1's complement add 1 to it to and you get 2's complement. So I am now going to add 2's complement of 0 and 1 add and this 1 plus 1 is equal to 0 with the one carry one. This extra carry is not required in this case, we are talking of 4-bit number here so you have to be careful, there is sort of a caution here.

I am dealing with 4-bit numbers A and B, A is 4-bits B is 4-bits so I am expecting a 4-bit result. it is not A plus B the result can be 5 bits, A is 4-bits, B is 4-bits so I am subtracting A minus B and the result has to be 4-bits so all you have to do is to discard the carry ignore it. I know that result is going to be 4-bits or less I discard the last bit. Now (Refer Slide Time: 27:09) this number is how much is this, so this is discard carry, this number 0111 is 7 so 12 minus 5 is equal to 7 I can also do it in 1's complement way but then I will get one less number and then add 1 to it finally.

Suppose I do 1's complement addition 1's complement of A B is, B is 0101 1's complement is 1010, B is 0101 and 1's complement of B is 1010 add it and it becomes 10110, now I can't discard this carry because I know that one carry has to be, of course this carry can be discarded I have to add 1 to it, I can discard this carry and add 1 to it or you can say sort of it is not logical this 1 is brought here and added which is called End Around Carry. You will see in the book it is called EAC.

(Refer Slide Time 29:00)



You bring the carry around and add it here, you have to add 1 to 2's complement number. You have to add 1 to this because it is 1's complement and you want 2's complement. The 2's complement gives this subtraction number. The subtraction of two numbers subtraction of B from A the result is the 2's complement addition of A and B. In other words A minus B is same as A plus 2's complement of B not 1's complement.

Since we need to have 2's complement and we have done only 1's complement I have to add anyway 1. Somewhere 1 has to be inserted and that 1 is inserted here but it's okay I you discard 1 and then add 1 as if you are bringing it back and adding it. I think it is more for people to remember, this is sort of a tip for remembering that 1 has to be brought so they say bring this and add but then you will find it a very serious thing you will say end around carry. You bring it back they will put an arrow like this and show it like this. If you want to do it fine I have no problem as long as you understand what is happening.

Now we are talking of a positive number and subtracted from the positive number the result is a positive number. Suppose I have a number from which I am subtracting is larger than the given number the result will be negative, we will also look at that possibility.

What will happen if the difference is negative? It happens when N is equal to 5. I will put it the other way 5 minus 12. Can I now say A is equal to 5, B is equal to 12 the result has to be minus 7. So I will do this A is equal to 0101, 2's complement of B by now we know how to do that is 0011 add 0011 to that you have to add 1 so what will be 2's complement of B? It will be 0100 that is because 1's complement is 0011, 2's complement is add 1 it becomes 001. A minus B is same as adding A and 2's complement of B which is 1001. By addition it is right, but the answer is not right. There is no carry here, significantly. There is a carry here there is 0 in the last digit. So whenever there is a 0 in last digit and a carry it represents the positive result, difference is positive. Whenever the difference is positive you will have a 0 in the last bit position which is the required bit position and this one which is going to be discarded there is a 1 in the position to be discarded and 0 in the last position of interest. Here it is exactly the opposite; I have a 1 here in the last position of interest and 0 in the position to be discarded. Hence this gives us a signal that this is a negative number. If the last position of interest is 0 and if there is a carry to be discarded then it's a positive result so the answer is right.

If on the other hand you get a 0 here and 1 here you know it's a negative number that means I have to get a 2's complement of that. The correct answer is not this the number is the 2's complement of the difference. So if I take the 2's complement of this number so I will call this 2's complement this is not A minus B really this is A minus B let us say star. if I take the 2's complement of this then this is going to be, 1's complement is 1, the 1's complement is 0110 add 1 to it I get 2's complement which is 0111 which is 7. But you should not take it as 7 because of this, this is minus 7. So when the number is negative you take the 2's complement and give the negative sign bit and when the number is positive you take the number as it is.

How do you know that it is a positive number or negative number? The discarded bit is 1 in the case of positive number and the last bit of the interest is a 0, the last bit of interest is 0 it is a positive number, this is called MSB. This is the most significant bit as far as we are concerned. This is the most significant bit and this is the least significant bit (Refer Slide Time: 34:56) I think we already mentioned this once. When you have several bits the lowest significant bit, the next I have is the maximum significance of this bit. So this is the MSB and this is the LSB. so if the MSB is 0 there is a positive number forget about discarding because anyway we are going to discard that 1 and 0, here the 0 is discarded and this one is discarded so let us not worry about, there is an extra signal but don't worry about that.

I have this most significant bit of 0 it's a positive number and here I have the most significant bit as 1 which is a negative number. Once the result is negative the 2's complement addition of B to A does not give you A minus B if the difference happens to

be a negative number indicated by the fact that MSB is 1. If the MSB is 1 the difference A minus B which you computed by adding these 2's complement of B to A is not the correct number it is the 2's complement of that difference which is the correct number which is the 2's complement. But you don't have to worry about it because this is a negative number and how do you say minus 7 in your computer? You add 7 you can put 7 binary 7 you can put binary any number, how do you put minus sign I said all symbols have to be represented by binary codes whether it is an alphabet, special symbols, comma, punctuation, email, letters, database, marks etc it is all binary numbers binary digits bits so negative has to be somewhere so that is our way of telling the computer that a negative number is also the same way the computer has a way of telling us that the negative numbers are also the same way.

Therefore, for any number of bits you define the number of bits in a number the most significant bit if it is 1 it is a negative number, if the most significant bit is 0 it is a positive number. Even though for our consumption, for our understanding to make sure I have done this arithmetic right I want to do the 2's complement and check whether it is really 7. I knew it's negative but I am not sure it is 7, how do I check it is 7? I check it by taking 2's complement and finding out indeed 7. But this is 0 but I know the number is negative so I cannot have 0 here if the number is negative, the answer is minus 7, when the answer is minus 7 I cant have a 0 here, I have 1 here. So negative numbers are represented in computers by 2's complement whose MSB is 1. Thus if you want to give a negative number input of the computer you don't give a number and punch in the negative signs in the calculator. Of course you try that but minus or sign of the number you have in the calculator the plus minus sign if you punch that key a minus does not go and get stored that minus is translated immediately you have to convert whatever number you have written earlier it has to be converted to 2's complement and stored with the one MSB.

So MSB 1 is indicative of the fact that a negative number is represented. Similarly the computer when it finds out the result gives you a negative number, in the case of negative numbers it gives you 1 as MSB you know it is a negative number you deal with it the way you want, it is the final step where you want to give your answers in the examination then you won't write it like this you will say it is minus 7 in your answer paper. But if this is going to be fed into the next stage of computation because after all you do computation a long equation or whatever arithmetic problem solving this is an intermediate result which you will feed into the next stage and at that time again you have to feed minus 7 to the computer and in the computer you don't have to worry about translating into minus 7 and making it plus 7 again making it minus 7 and do all that stuff. So this is minus 7. This extra step was just to tell you it is not wrong arithmetic that we did. I showed you that right arithmetic is all in right arithmetic only. So the answer is this in the case of 5 minus 12 and this the case of 12 minus 5 (Refer Slide Time: 39:37). Here for the 12 minus 5 the answer is plus 7 so it is 0111 so 0111 is plus 5 or 0111 is plus 7 so now this 12 minus 5 is plus 7 0111, 5 minus 12 is equal to minus 7 and if you don't believe me all I am asking you to do is to take this 2's complement and see for yourself it is all I am saying.

Any questions here? The complementing of numbers makes subtraction and adding operation that is where the crux of the whole matter is today.

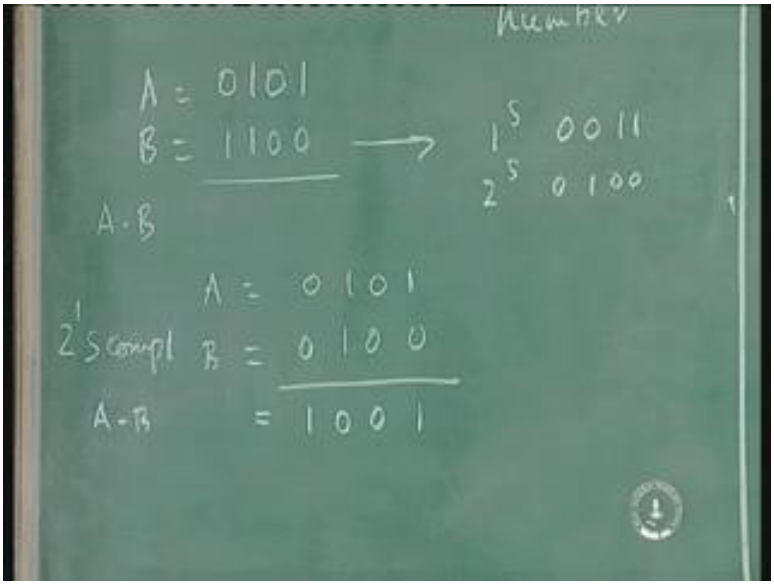
I introduced number systems the complement number system to tell you, of course I have always told you this is the way to do arithmetic where I thought I want to tell you why they do it. Why they do it is 2's complement numbers if you have you can add and then get a subtraction. Subtraction has converted the reducing operation. Why do we have to do that? I can have a single hardware called adder one hardware unit and since I am not going to use the same hardware both for adding and subtracting at the same time when I want to add I add and when I want to subtract I subtract.

All I have to feed is the numbers in 2's complement form. If I feed the numbers with regular form it gets added and in 2's complement form for if you put it then it gets subtracted. So the question now is how do you convert a given number to 2's complement number that is extra logic. As I said nothing is free, there is no such thing as free lunch, somebody invites you for lunch you know that there is something else is coming in favor.

That means you need to do something to get the 2's complement numbers. But it is easy because al you have to do is inversion and add 1. How do you get an inverter in the number just put a positive inverter. Thus the quickest way to invert is to put an inverter which all of us know so first I will tell you how to go to 1's complement then we will see how to do 2's complement of a given number.

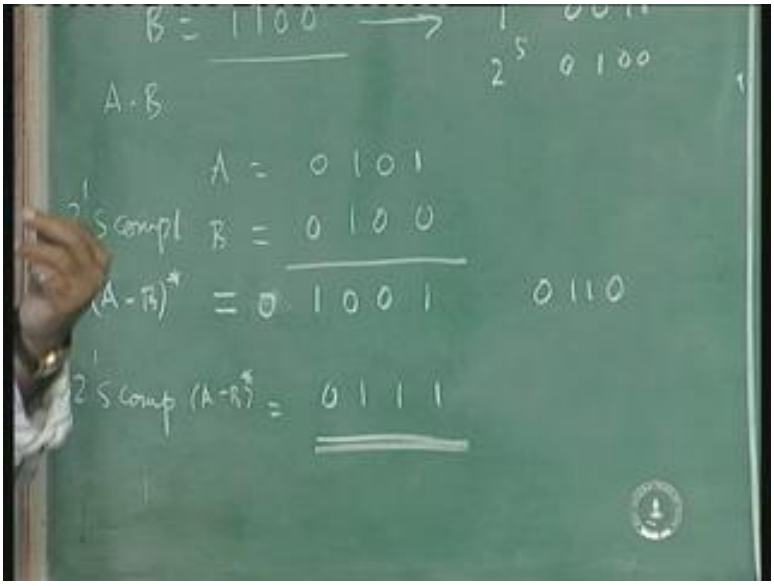
I have now reduced the subtraction to 2's comp addition, you all agree on this? If there is anyone who doesn't agree on this, it is better. The next step is how we are going to convert a given number to 2's complement before I can add it. So, first let us do 1's complement that is why I told you that by hardware-wise it is easier to do a 1's complement and plus 1. Even though in principle you want to do this, this borrowing from the previous thing and all those things I want to avoid a simple complementing will give you 1's complement all I have to do is add an extra 1 somewhere, I can do it in the beginning or do it in the end.

(Refer Slide Time 32:17)



So 1's complement is an inverter so if it is A A bar no problem, any bit is ok, but the problem is a given bit A can be converted into 1's complement with inverter but I don't know when to convert when not to convert.

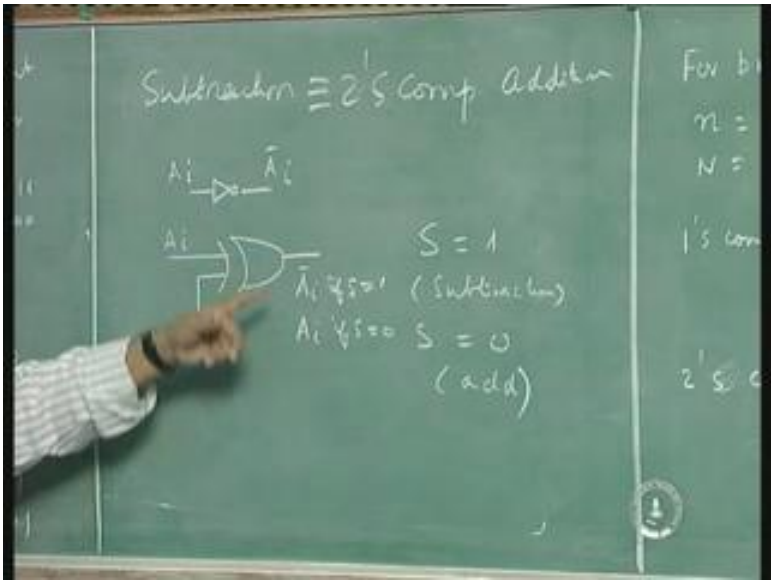
(Refer Slide Time 34:31)



When it is the adding operation I have the same hardware unit called adder in which I give A and B and without converting it adds up, and when I give A and convert B as 2's complement and feed it, it subtracts so I need to know when to give the number directly as a correct number, when to give this number as a 2's complement number. Whenever I need subtraction I need to give the 2's complement number and whenever I need addition I need to give it directly so I need to have a control of this addition the converter. I need to have a control of this inversion operation whenever I need so it is a very simple technique for that.

Let us do the Exclusive OR gate. Suppose I have an Exclusive OR gate this is the number I want to invert A_i not always but whenever I want, when do I want is when S is equal to 1 S is the subtraction. Suppose I give it by S here the second input of A if this is 1 and this is 1 what is the output of Exclusive OR gate? 1 1 is the Exclusive OR input output 0. When this is 0 and this is Ai this also becomes Ai.

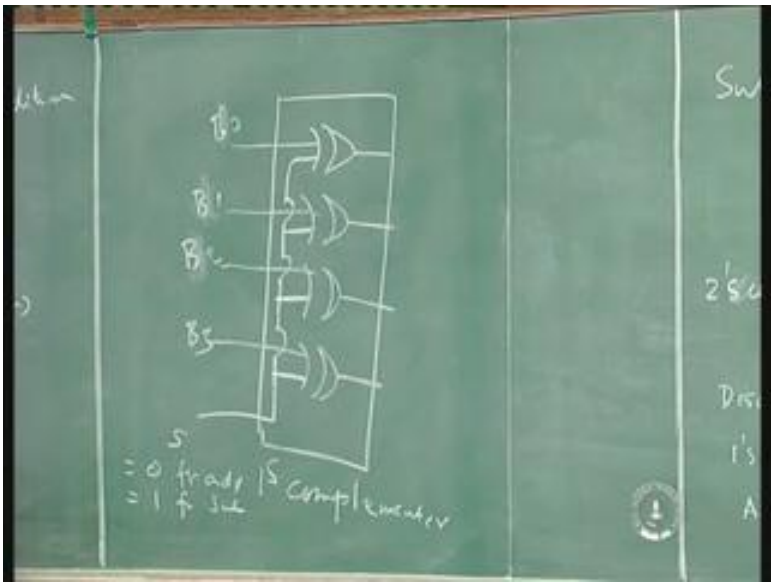
(Refer Slide Time 45:39)



When S is 0 that's for add, when I give S is equal to 0 it is addition operation, this is $A_i$bar (Refer Slide Time: 45:14) if S is equal to 1 A_i is this. If S is 0 this is 0, this is 0, this is 1, and this is 1, S is 1, if this is 0, this is 1, this is 1, this is 0. So whenever I want to invert I give S is equal to 1 whenever I do not want invert I put S is equal to 0. Now not only I can invert and get 1's complement but I can do it at nil, do it when I need it.

So my simple 1's complement circuitry is a bank of Exclusive OR gates to which I feed the inputs so this is my 1's complement circuit or I will call it 1's complement because you want to give big names to small things.

(Refer Slide Time 47:38)



My 1's complementer hardware I give a series of Exclusive OR gates let us say 4-bits let us be happy with 4-bits today B_0 B_1 B_2 B_3 just can be anything for the sake of continuity of the lecture I am giving Bs so I am converting B. It can be A, can be P, can be Q it doesn't matter.

Now I am going to connect all the other inputs together and give this as S. Whenever I want subtraction I make this S is equal to 1 I get B bar B_0bar B_1bar B_2bar B_3bar. Whenever S is 0 that means I add S is equal to 0 for add, 1 for subtract so if it is adding it is A_0 A_1 A_2 A_3 will come out and for subtraction A_0bar A_1bar A_2bar A_3bar will come out. Therefore it is a 1's complement of circuitry of course I have to add 1 to it so we will see how to do that.

So all I have to do now is I have an adder which we did elaborately the last couple of lectures and I now have a 1's complementer which is a simple logic and also proved to you that 1's complementer or 2's complementer can be used in conjunction with an adder to get a subtraction then all have to do is to glue it together put it together and get a subtractor, we will see that in the next lecture how to get the subtraction. Now there is nothing you can do it yourself. I have an adder, I have a complementer what else you need, tell me where will you give that extra 1. ''Carry in'' is usually 0 in adder, the first stage 'Carry in' is always 0 you make that 1 not always 1 but whenever you want subtraction. So you make that same as S. So I have a full adder A directly fed through this complementary circuitry and connect C to the same S which is going to control the complementary circuitry then output I get, fair enough, very simple. All I have to do is the drawing part of it. You can do it yourself, it is in the book. You just have to draw this

complementer, direct input, complemented input, S controlling the complementing as well as the ‘Carry in’ put to get the output result as negative A minus B.