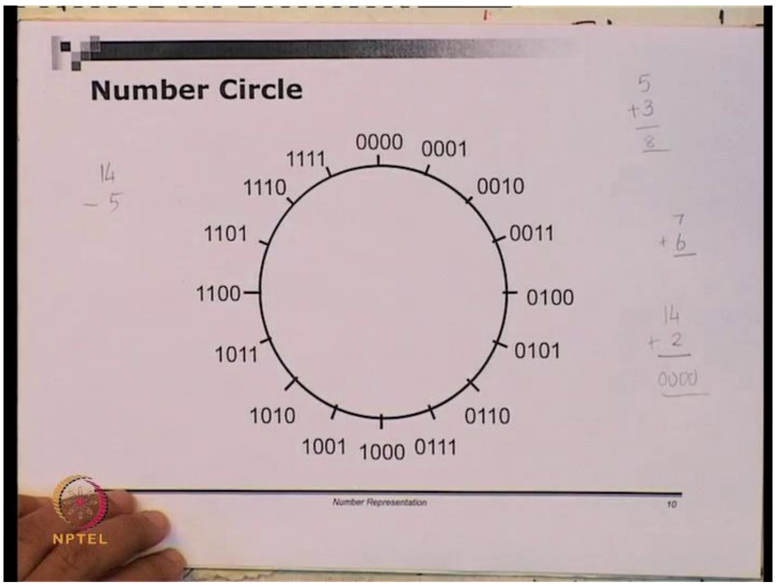
Digital Circuits and Systems

Prof. Shankar Balachandran
Department of Electrical Engineering
Indian Institute of Technology, Bombay
And
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Module – 49
Addition and Subtraction in 1's and 2's Complement Form

Welcome back, so in this module we are going to look at Addition and Subtraction in 1's complement and 2's Complements Form. So, before we go in to the actual mechanics of 1's complement and 2's complement. I want you to go back and look at this number circle.

(Refer Slide Time: 00:31)



Number Circle

So, let us assume that these numbers as before where only... So, theses are 4 bit values let me just assume that we are looking at unsigned interpretation of this. So, this will be number 0, this will be number 15, this is number 11, this is number 3, and so on, let us keep it as it is. Now, if I ask you to do addition, let us say I ask you to add 3 to 5, I want you to add number 3 to number 5. So, in the number circle you can do something very interesting, you start with the number representation for 5.

So, what is the representation for 5? It is 0 1 0 1, if you know that and if you have to add 3 you go 3 steps forward 1, 2, 3; that is the number, that is... So, this 1 0 0 0 is the binary representation for 5 plus 3, we know that is 8, let us take another example. Let us say I want to do 7 plus 6, so I need to start with the binary representation for 7, this is an unsigned circle. So, I start with the binary representation which is 7, 0 1 1 1; I go 6 steps forward from here 1, 2, 3, 4, 5, 6, 1 1 0 1. So, 1 1 1 0 actually stands for number 13 we know that, so this is a good thing. In the number circle, if I start from a number and I take so many steps forward, I will end up with some other positions in the number circle and that number circle gives me the representation for the result that I am looking for. There are a few problems; however, for example, let us say I ask you for 14 plus 2, then you would start from 14 which is 1 1 1 0 and you go 2 steps forward, you end up with 0 0 0 0.

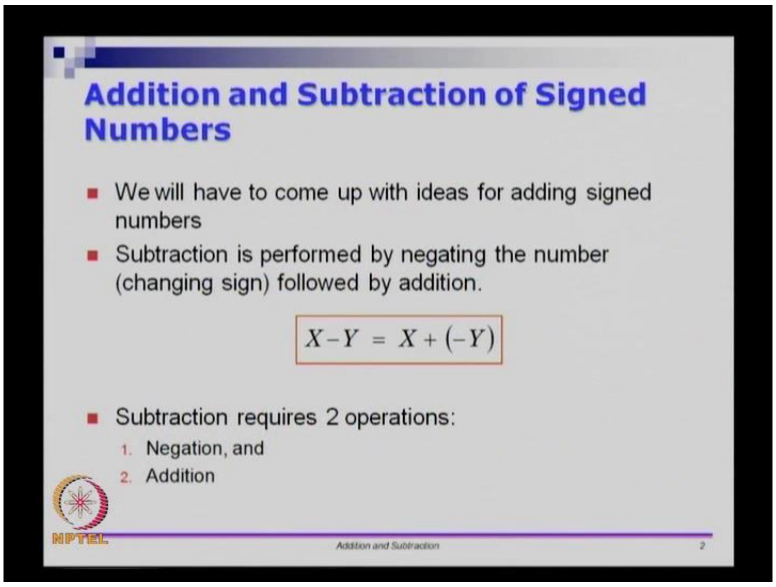
So, if you say that 14 plus 2 is 0 0 0 0, it is technically not correct, 14 plus 2 in decimal is 16; however, since I am forcing you to give only a 4 bit representation, even in the unsigned numbers you cannot represent 16 using only 4 bits. So, this is the first wakeup call that you should all have, that in any representation if you restrict the number of bits you will not be able to represent all the numbers, which means addition and subtraction and whatever you do may result in something which is not correct. So, the result that you get may not be correct.

Because, you are adding numbers which are exceeding the range in which things can be represented. So, I want to do something else, so I want to do something like this, let us say I want to do number 14 minus number, let us say 5, I want to do subtraction. So, if want to do subtraction one thing you can do is, you can start from here 14 and go 5 steps in the counter clockwise direction. So, you start from 14, go 5 steps backward 1, 2, 3, 4, 5 that is 1 0 0 1. So, 14 minus 5 is 1 0 0 1 not 9, so which is correct.

So, let me try another example, let us say I do 5 minus 4, then 5 is 0 1 0 1 if I go 4 steps backward 1, 2, 3, 4 the result is 1. So, we are good, this just like addition we can end up with small problems. So, for example, if I start with number 2 and I subtract 5 from it, I start with 2, and then I have to go 5 steps backward 1, 2, 3, 4, 5. So, it looks like 2 minus 5 is minus 13, 1 1 0 1 is 13. So, 2 minus 5 it actually gives plus 13 as the result not minus 13.

So, in the unsigned representation you cannot event represent negative numbers, so 2 minus 5 may end up with 30. So, this is something that is wrong, but given that we can represent only using so many, we cannot represent negative numbers we cannot do anything about it at all, so no negative numbers can be represented. So, just keep this in mind, because we will need this kind of interpretation for the other once also. So, now let us jump forward, if I want to do addition and subtraction of signed numbers.

(Refer Slide Time: 04:55)



Addition and Subtraction of Signed Numbers

- We will have to come up with ideas for adding signed numbers
- Subtraction is performed by negating the number (changing sign) followed by addition.

$$\boxed{X - Y = X + (-Y)}$$

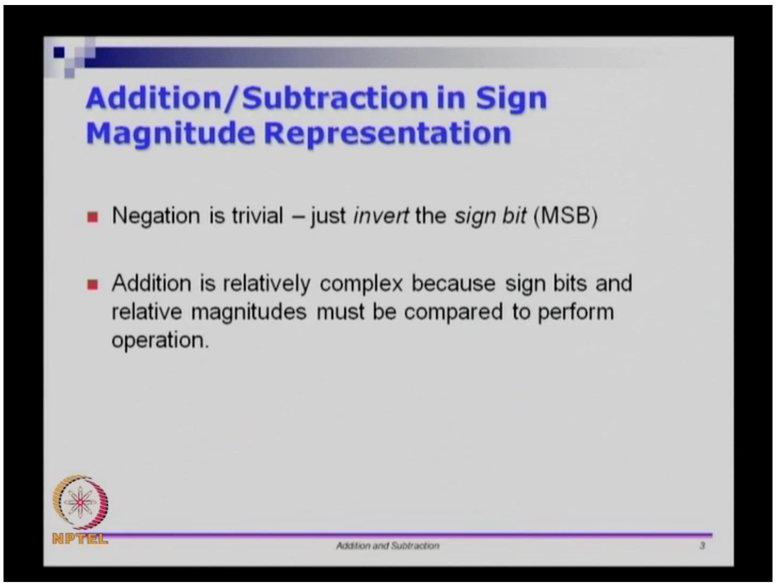
- Subtraction requires 2 operations:
  1. Negation, and
  2. Addition

Addition and Subtraction 2

When I say signed, it could be sign magnitude 1's complement, 2's complement it could be any of those, we will have to come up with some idea for adding the sign numbers first. So, addition and subtraction are two different operations, so we need to come up with an idea for adding numbers. But subtraction becomes slightly easy, what you have to do is, subtraction is usually performed as two steps. So, if I want to do something like this, let us say I want to subtract Y from X, then I can as well do this, I can do it as X plus minus of Y.

So, as long as I take Y which is a positive number, which is some number. So, I have X and Y are two different numbers, if I take Y and negated and if I add the negated result to X that is equivalent doing X minus Y. So, that is the first thing that I want you to note. So, if I want to subtract number Y, it could be positive or negative from number X which could again be positive or negative, all you have to do is take Y, negate it, added to X. So, this is a key thing that we will do over and over for all the representations later.

(Refer Slide Time: 06:08)



Addition/Subtraction in Sign Magnitude Representation

- Negation is trivial – just *invert* the *sign bit* (MSB)
- Addition is relatively complex because sign bits and relative magnitudes must be compared to perform operation.

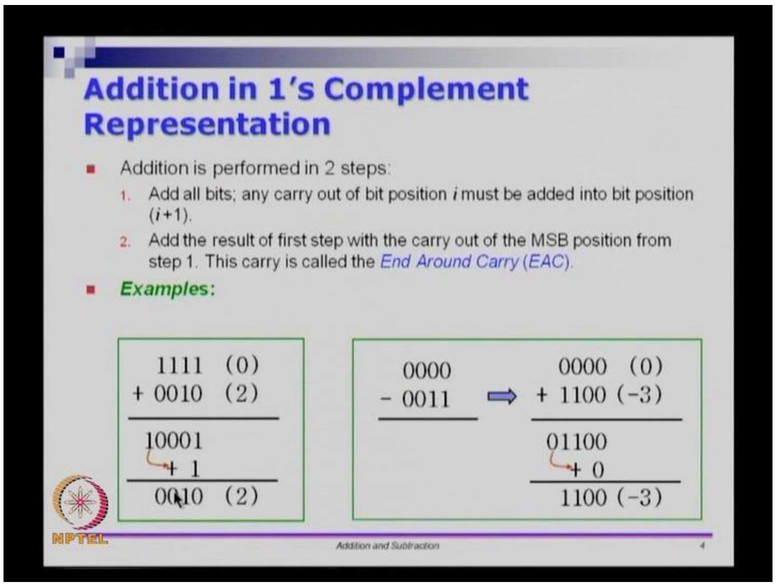
NPTEL
Addition and Subtraction
3

So, let us start with the very first a sign representation, namely the sign magnitude representation. Addition and subtraction in sign magnitude representation, negation is actually trivial. So, if you want to negate a number, all you have to do is take the most significant bit and flip it, so if it is positive it will go to negative, if it is negative it will go to positive that is. However, addition is relatively complex, because addition is complex, subtraction also becomes complex, because subtraction is negation followed by addition.

So, addition it is complex, because sign bits and relative magnitudes must be compared to perform the overall operation. If I give you something like add 0 1 1 with 0 1 0, then the result can actually be a problem, if you end up with 1 0 0 and if you leave it as it is, it saying that 3 plus 2 is not 5, but it is telling you that the number is 1 0 0, which is negative 0. So, this is the minor problem, the more complicated problem is, if I give you numbers like plus 5 and minus 3 you want to add them and so on, it is not easy to do this using the sign representation.

So, this is the reason why sign magnitude representation is not useful in computers. This was the first representation that people came up with and later they realize that, it is not really nice, because addition becomes a very complicated process.

(Refer Slide Time: 07:37)



Addition in 1's Complement Representation

- Addition is performed in 2 steps:
  1. Add all bits; any carry out of bit position *i* must be added into bit position (*i*+1).
  2. Add the result of first step with the carry out of the MSB position from step 1. This carry is called the *End Around Carry (EAC)*.
- ***Examples:***

$$\begin{array}{r l} 1111 & (0) \\ +\,0010 & (2) \\ \hline 10001 & \\ +\,1 & \\ \hline 0010 & (2) \end{array}$$

$$\begin{array}{r} 0000 \\ -\,0011 \\ \hline \end{array} \Rightarrow \begin{array}{r l} 0000 & (0) \\ +\,1100 & (-3) \\ \hline 01100 & \\ +\,0 & \\ \hline 1100 & (-3) \end{array}$$

Addition and Subtraction 4

So, let us see how to do addition in 1's complement representation, addition itself is done in 2 steps, first one is you add all the bits. So, if I given two numbers which are already in 1's complement form, so that is the key thing. When we say addition in 1's complement representation, the two numbers are given in 1's complements form, the result is expected in 1's complement form, the question is, how do we do it. So, we know what the input and output requirements are, but we do not know how to do it.

So, I am explaining you the process of how to do it, if I want to add two 1's complement numbers, it by itself is done in 2 steps. So, the first step is you add the bits as it is, you treat the numbers as though they are unsigned numbers and do the addition as it is. So, we will talk about how to do unsigned addition later, but this is also something that I have given you as homework. I told you to add two 4 bit numbers by designing half adder, full adder and 4 bit adder, you already done that.

So, to add 2 numbers in the unsigned form, it is relatively easy. What you are going to do for 1s complement is, treat the numbers as though they are unsigned, do addition first. So, add all the bits, carry out of the bit position i must be added into the bit position i plus 1, this is just like regular decimal. If I add two digits at the 100th position, the carry that is coming out will go to the 1000th position and so on, it is a same thing.

The key thing is, add the result of the first step, which means the left most one with the carry out of the MSB position from step 1. So, this is called End Around Carry, so I will show in example of what this means and it will make it clearly. So, let us look at let us say 1 1 1 1, if I want to take 1 1 1 1, 1 1 num 1 1 in the 1’s complement representation is number 0, this is actually negative 0.

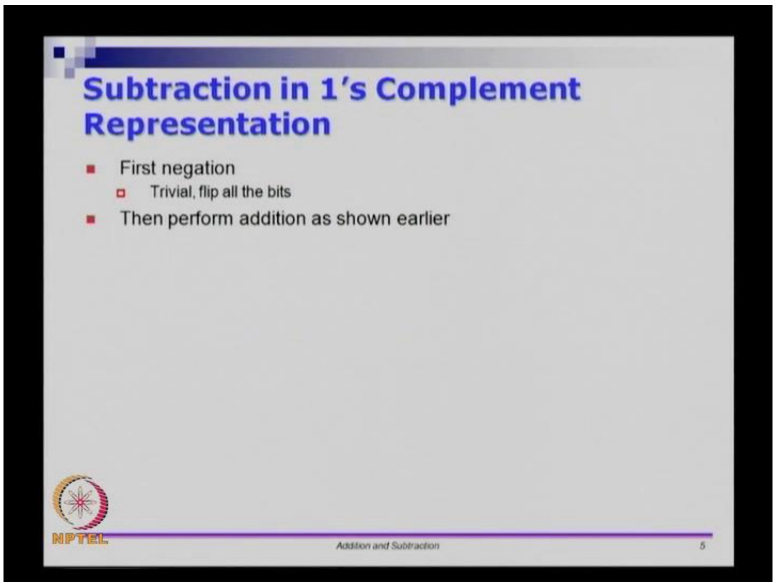
And if I want to add 0 0 1 0 which is 2, then the way to add that is, you add, you ignoring the fact that they are 1’s complement numbers, just add them as it is. So, 1 plus 0 is 1, 1 plus 1 is 0 carry over 1, 1 plus 1 is 0 carry over 1, 1 plus 1 is 0 carry over 1. The ends state carry you take it and added as the last position at this right most position, you add it once more and when you do that you get this result, if this results in a carry now, do not do anything.

So, for once add the numbers if there is a carry take that carry and add it once more and whatever result you get in the n digits, n bits is your final result. If this second addition gives you another bit ignore it, do not do anything with it, so this is what the 1’s complement addition is. If you want to do 1’s complements subtraction, in this case this 0 0 0 is negative 0 and we want to subtract 3. So, 0 0 1 1 is 3, you want to subtract 3 from 0, we know that the result is minus 3, let us see how to do that.

So, if we are doing subtraction the first thing we do is, whatever this is right we keep it as it is and I said minus of a number we will treated as though it is. So, this is negation of a positive number, so this is x minus y will treated as x plus minus y and how do you get minus 3, minus 3 is just flip all the bits of plus 3. So, it is 1 1 0 0 again you add it as before, in this case the end carry is 0, you add the end carry, the result is 1 1 0 0. 1 1 0 0 if you go back and interpret it in 1’s complement form it is negative 3, I want you to quickly do that before we go to the next slide.

So, 1 1 0 0 I want you to convince yourself that it is minus 3, the way to do that is note down the most significant bit, if it is 1 the result is negative. So, it is negative, flip all the bits, so flip all the bits 0 0 1 1, 0 0 1 1 interpreted in unsigned form is 3. So, it is negative 3, just convince yourself that this is the case.

(Refer Slide Time: 11:51)



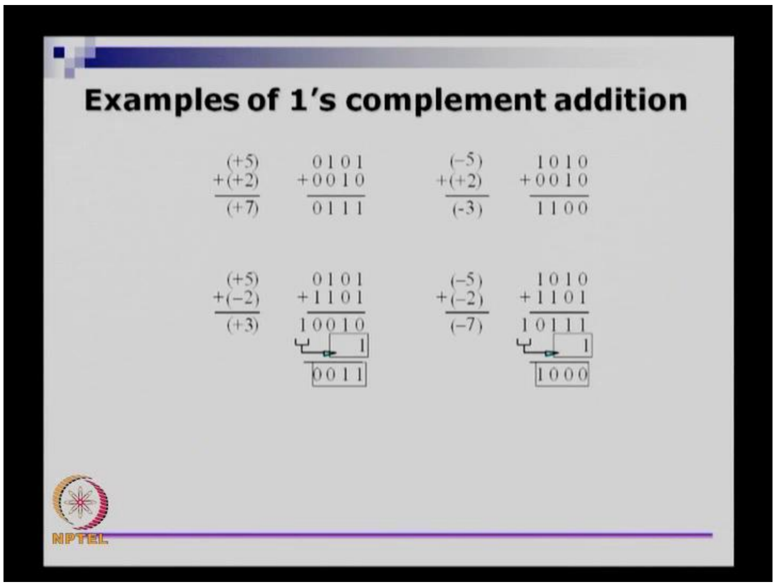
Subtraction in 1's Complement Representation

- First negation
  - Trivial, flip all the bits
- Then perform addition as shown earlier

Addition and Subtraction 5

Subtraction in 1's complement is you first go and negate, because it is trivial to do and then you perform the addition as I showed earlier. So, it is fairly straight forward process.

(Refer Slide Time: 12:02)



Examples of 1's complement addition

$$\begin{array}{rr} (+5) & 0101 \\ +(+2) & +0010 \\ \hline (+7) & 0111 \end{array} \qquad \begin{array}{rr} (-5) & 1010 \\ +(+2) & +0010 \\ \hline (-3) & 1100 \end{array}$$

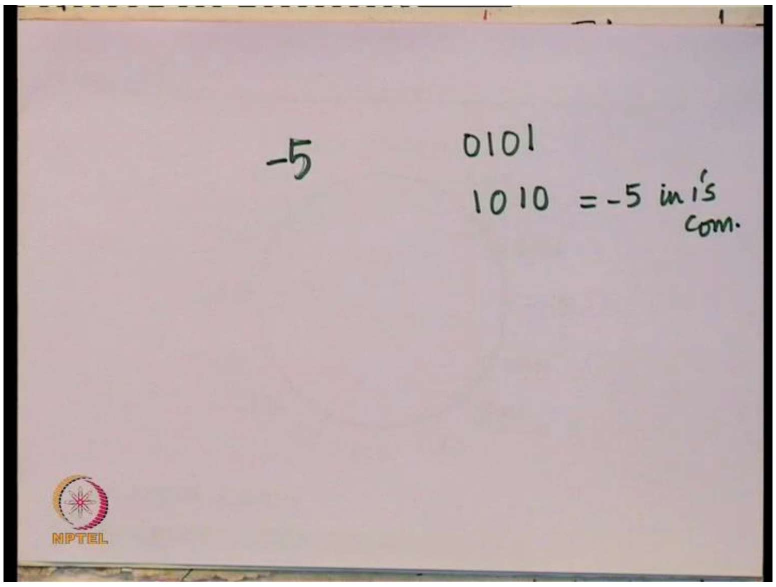
$$\begin{array}{rr} (+5) & 0101 \\ +(-2) & +1101 \\ \hline (+3) & 10010 \\ & \boxed{1} \\ \hline & \boxed{0011} \end{array} \qquad \begin{array}{rr} (-5) & 1010 \\ +(-2) & +1101 \\ \hline (-7) & 10111 \\ & \boxed{1} \\ \hline & \boxed{1000} \end{array}$$

So, let us see several examples of 1's complement addition, so let us say I want to do this. I want to add positive 5 with positive 2, I know that the result must be positive 7 and in all these cases, I am go to assume that the numbers are represented using 4 bits. So, how do you represent plus 5 in 4 bits in 1s complement, it is 0 1 0 1. So, plus 5 the Titus representation requires 3 bits it is 1 0 1, but I said we will do all the operations in 4 bits.

So, the sign is 0 1 0 1, so 0 1 0 1 is the 1's complement interpretation would give you plus 5. How do you represent plus 2, it is 0 0 1 0 and I want you to do add this. So, now we will forget the fact that you are actually adding 1's complement numbers, we will directly add them as though they are unsigned numbers. When you add them, the unsigned numbers give you 0 triple 1 and the carry out that is given in 0. So, there is nothing further to add, the result is 0 1 1 1 itself.

So, if you take 0 1 1 1, interpret that in the 1's complement form, you will see that it is plus 7 itself, let us do this next set minus 5 added to plus 2. So, how do you represent minus 5, so for minus 5 you should take the representation.

(Refer Slide Time: 13:38)



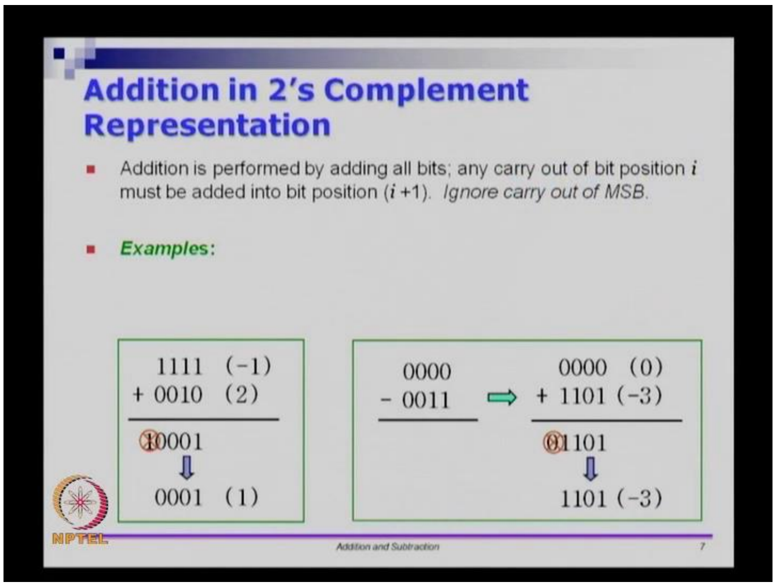
So, let me do that on a piece of paper, minus 5 if I want to represent minus 5 as a 1's complement number, I first take 5 which is 0 1 0 1 and I flip all the bits around. So, that is 1 0 1 0, so 1 0 1 0 is actually minus 5 in 1's complement form. So, we have that and I want to add plus 2 to it, so plus 2 is 0 0 1 0, so let... So, minus 5 is 1 0 1 0 and plus 2 is 0 0 1 0, I add both of these, it results in 1 1 0 0.

So, for this 1 1 0 0 I want you to remember that 1 1 0 0 is not generated any carry. So, because I added 1 and 0 which is a 1 it did not any generate carry nothing else to do, if you go and interpreted that you will see that it is minus 3, let us look at this one 5 plus minus 2. So, this 5 plus minus 2 we know that it should be plus 3, let us see how this will work, plus 5 is 0 1 0 1 and minus 2. So, 2 is 0 0 1 0, so minus 2 should be 1 1 0 1 you add these two numbers.

When you add these two, we are generating some carry outside, we take this carry added to this and that will result in 0 0 1 1. So, we are taking this carry is as good as removing it and adding it to the last stage and you put this together with 0 0 1 1. So, we know that plus 5 added to minus 2 is plus 3, so we are still doing addition here and if you want to do minus 5 plus minus 2 minus 5 is 1 0 1 0. We know we have done this here minus 5 is 1 0 1 0 and minus 2 is 1 1 0 1, we add these two together and this case also generates a carry.

This case is generating a 1 and you add the one that gives you 1 triple 0 and we know that 1 tripe 0 interpreted in 1’s complement form is negative 7. So, this is the result of 1’s complement addition.

(Refer Slide Time: 15:53)



Addition in 2’s Complement Representation

- Addition is performed by adding all bits; any carry out of bit position *i* must be added into bit position (*i* +1). *Ignore carry out of MSB.*
- **Examples:**

$$\begin{array}{r} 1111\ (-1) \\ +\ 0010\ (2) \\ \hline \boxed{1}0001 \\ \Downarrow \\ 0001\ (1) \end{array}$$

$$\begin{array}{r} 0000 \\ -\ 0011 \\ \hline \end{array} \Rightarrow \begin{array}{r} 0000\ (0) \\ +\ 1101\ (-3) \\ \hline \boxed{0}1101 \\ \Downarrow \\ 1101\ (-3) \end{array}$$

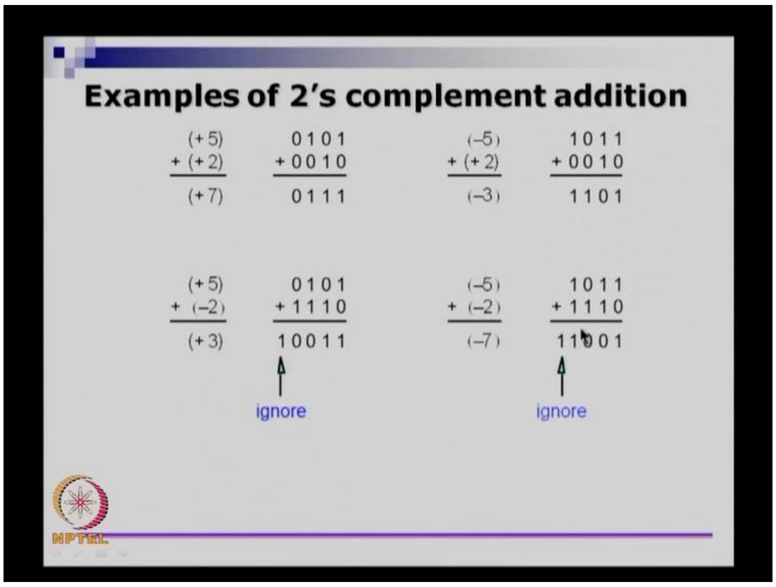
Let us do addition in 2’s complement representation before we go to subtraction. So, addition is performed by adding all the bits as though they are unsigned numbers, any carry out from one position should go to the next position and the last stage carry can be ignored completely. So, in the 1’s complement we took the last stage carry and we added it back, in 2’s complement when you add two numbers the last stage carry you do not have to do anything with it.

So, let us look an example, so if I take 1 1 1 1 and I add 0 0 1 0 to it, so let us forget what the values for a while, if I tell you that this is 2s complement and this is 2’s complement I want, you add you just add these as it is this generates the carry 1 you ignore it take only the last 4 digits and you are put that as the result, let see whether this is correct first of all 1 1 1 1 is minus 1 and 0 0 1 0 is 2 in the 2’s complement representation, the results seems to be 0 0 0 1 if you interpret that 2’s complement the result is 1.

So, it is consistent if I want to do subtraction, so in this case we are subtracting plus 3 from 0 we want the result to be minus 3. So, to do that you take 0, 0 this is subtrahend and this is minuend, subtrahend you keep it as it is the subtrahend is what you subtracting from and minuend is what subtracting. So, if I do a minus b a is a subtrahend and b is the minuend, so subtrahend we keep it as it is and for the minuend.

So, we are doing minus b, minus b is same as plus of minus b, so we take this which is 0 0 1 1 I know that number interpreted in 2’s complement is 3, I go and look at the 2’s complement representation of minus 3 that seems to be 1 1 0 1 you add these two together and the result is 0 1 1 0 1, again you ignore the carry and that gives you 1 1 0 1. So, there is a result is minus 3.

(Refer Slide Time: 18:05)

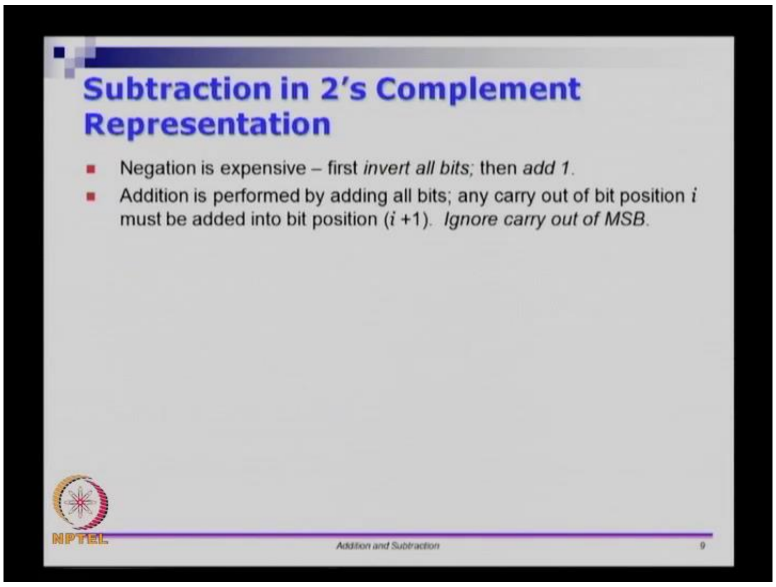

Examples of 2’s complement addition

```
   (+5)      0101           (−5)      1011
 + (+2)    + 0010         + (+2)    + 0010
 ------    ------         ------    ------
   (+7)      0111           (−3)      1101


   (+5)      0101           (−5)      1011
 + (−2)    + 1110         + (−2)    + 1110
 ------    ------         ------    ------
   (+3)     10011           (−7)     11001
            ↑                        ↑
          ignore                   ignore
```

Let see more examples here, there are several examples, so let us look at plus 5 and plus 2. So, plus 5 is 0 1 0 1 plus 2 is 0 0 1 0 I add these together it is plus 7, so that is straight forward, minus 5 is 1 0 1 1 in 2's complements. So, this is slightly different from 1's complement, so this is 1s complement plus 1 1. So, in 1's complement should have been 1 0 1 0 in 2's complements minus 5 is 1 0 1 1.

So, you take 1 0 1 1 plus 2 is 0 0 1 0 add these two together the result is 1 1 0 1, there is the carry that is 0 that is the generated the ignore it anyway, whether it 0 or 1 we ignore it if we want to do 5 plus minus 2. So, this is not subtraction of 2 from 5, we are adding minus 2 to 5, so plus 5 is 0 1 0 1 minus 2 is 1 1 1 0 you add these two together there is a carry that is generated you ignore the carry. So, the result is 0 0 1 1 or 3 the same thing with minus 5 and minus 2.

(Refer Slide Time: 19:11)



Subtraction in 2's Complement Representation

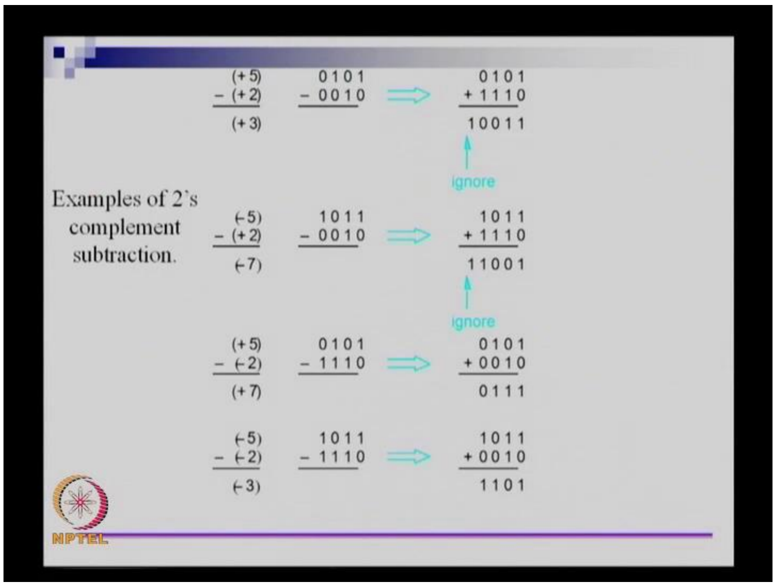
- Negation is expensive – first *invert all bits;* then *add 1*.
- Addition is performed by adding all bits; any carry out of bit position *i* must be added into bit position (*i* +1). *Ignore carry out of MSB*.

NPTEL Addition and Subtraction 9

Let us look at subtraction in 2's complement representation, so the subtraction in 2's complement is going to be negation followed by addition, we know that the negation in 2's complement is slightly more expensive. So, what we have to do is, we have to first invert all the bits and then had one to it that is how the we get the 1's complement interpretation, we take a 2's complement interpretation of a number, this could be positive or negative, if I want a negated I flip all the bits and add 1.

So, that is the way we do negation of 2's complement numbers, so again I want to emphasis that negation does not mean it works only on negative numbers, negation of any number in 2’s complement form is, you invert all the bits and add 1 that will give you the 2’s complement representation of negative x you start with x and addition is perform as it was before you ignore the carry.

(Refer Slide Time: 20:06)



Examples of 2’s complement subtraction.

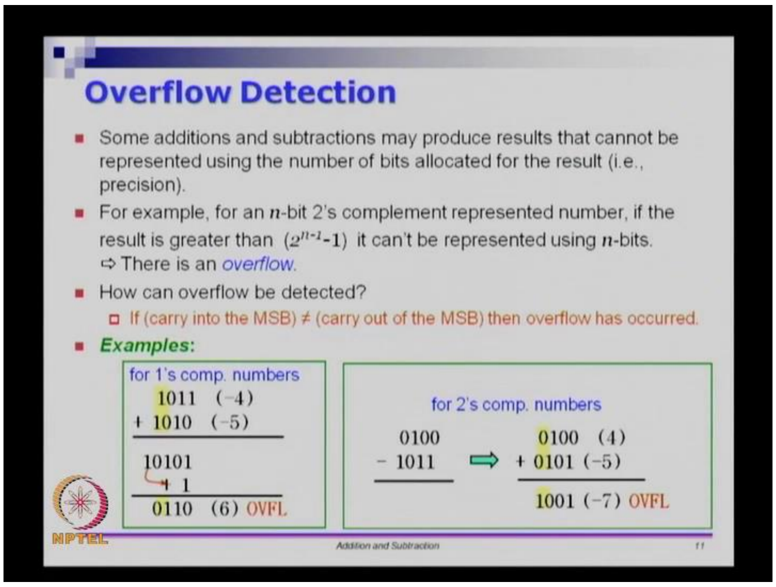
```
  (+5)      0101            0101
- (+2)    - 0010    ⇒     + 1110
------    ------          ------
  (+3)                     10011
                           ↑
                           ignore

  (-5)      1011            1011
- (+2)    - 0010    ⇒     + 1110
------    ------          ------
  (-7)                     11001
                           ↑
                           ignore

  (+5)      0101            0101
- (-2)    - 1110    ⇒     + 0010
------    ------          ------
  (+7)                      0111

  (-5)      1011            1011
- (-2)    - 1110    ⇒     + 0010
------    ------          ------
  (-3)                      1101
```

So, let see the examples, so I want to do 5 minus 2 that if I want to do 5 minus 2, why really want is 0 1 0 1 minus plus to is 0 0 1 0 that is what I really want, what I instead do is I add negation of 2. So, negation of plus 2 is minus 2 minus 2 is 2’s complement representation is 1 1 1 0 I take the 2’s complement interpretation of the subtrahend. So, I take the 2’s complement interpretation as it is for the subtrahend, for the minuend I have converted that are I have negated it and I have added it together and you ignore this bit 0 0 1 1 if interpret that 2’s complement that is number 3.

So, let us look at this example I want to subtract 2 from minus 5, so minus 5 is subtrahend for minus 5 the 2’s complement representation is 1 0 1 1 and plus 2’s complement representation is 0 0 1 0. However, I want subtraction of plus 2 for subtraction of plus 2 I do it as addition of minus 2. So, this is addition of minus 2 which is 1 1 1 0 here you add these and ignore this you get a result which is 1 0 0 1. So, 1 0 0 1 I want you to go and convince yourself that this stands for negative 7.

So, this easy to see because it is 1 0 0 1 it is starting with 1, so it is a negative number take 1 0 0 1 flip it 0 1 1 0 add 1 0 1 1 1 and that is 7, so it is negative 7. Then, it is look at these 2 plus 5 minus, minus 2 I want to subtract minus 2 from plus 5. So, plus 5 is 0 1 0 1 itself minus 2 is actually 1 1 1 0, but I want the negation of 1 1 1 0 which means it is add the negation of 1 1 1 0 which is 0 0 1 0 add these two together that gives you 7. And similarly I want to do go and work out minus 5 minus, minus 2.

(Refer Slide Time: 22:16)



Overflow Detection

- Some additions and subtractions may produce results that cannot be represented using the number of bits allocated for the result (i.e., precision).
- For example, for an n-bit 2's complement represented number, if the result is greater than $(2^{n-1}-1)$ it can't be represented using n-bits.
  ⇨ There is an *overflow*.
- How can overflow be detected?
  - If (carry into the MSB) ≠ (carry out of the MSB) then overflow has occurred.
- ***Examples:***

for 1's comp. numbers

```
  1011  (-4)
+ 1010  (-5)
-----------
 10101
    +1
-----------
  0110  (6) OVFL
```

for 2's comp. numbers

```
  0100          0100  (4)
- 1011   ⇨    + 0101 (-5)
------        -----------
                1001 (-7) OVFL
```

Addition and Subtraction 11

So, when you do this it is possible that they can be overflow, some additions and subtractions can produce over flow that cannot be represented in the range that is given. So, for example, if I give you a n bit 2's complement number, if the result is greater than 2 power n minus 1 minus 1 it cannot be represented using n bits. So, 4 bit 2's complement representation cannot represent 8 and above.

So, it is possible that when you do your operation addition or subtraction, it may end up with and an overflow which means it is ending with the number which is not represented within the range and many times is useful for us to know that the resultant operation is a overflow. So, how do you detect that if the carry into the MSB is not the same as carry out of the MSB then overflow is occurred, this is true both for 1's complement and 2's complement numbers.

Let see examples, so let us say I have 1 0 1 1 and I am adding 1 0 1 0 do it. So, 1 0 1 1 is minus 4, 1 0 1 0 is minus 5 adding minus 4 and minus 5 will give me minus 9, minus 9 cannot be represented using 4 bits that is the first think you have to remember let see how we can detect something like that. So, 1 0 1 1 added to 1 0 1 0, so if I want to just add I will add these two numbers that gives me the 5 digit value, you take the 1 add it around, this is the n carry and that gives you 0 1 1 0.

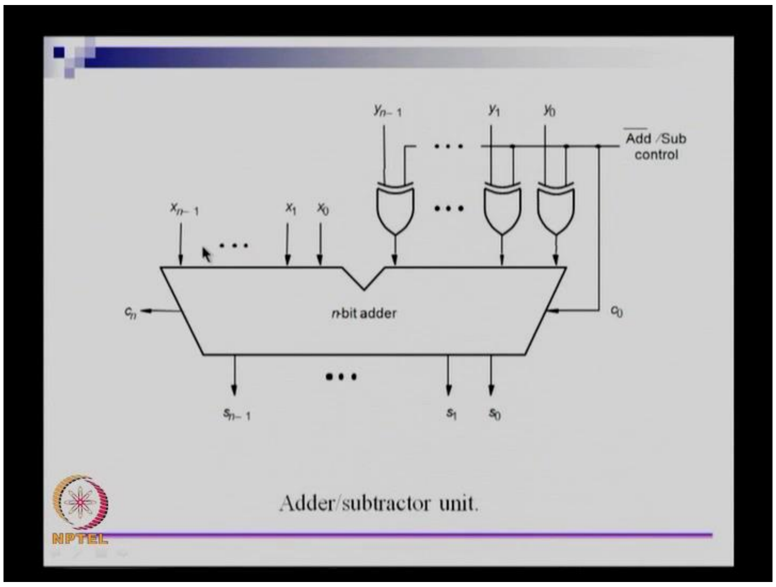
So, if I directly interpret 0 1 1 0 in 1’s complement form is positive 6 whereas, minus 4 plus minus 5 is actually negative 9, the only thing that indicates that this is not correct is let us look at the carry from this stage. So, the carry from adding 0 to 0 in give a 0 to the carry of 0. So, the incoming carry into the most significant bit stage is 0, the outgoing carry is 1 plus 1 is a 0, the carry out is a 1.

So, the outgoing carry is not the same as the incoming carry which means there is a overflow. Let us look at it for 2s complement number, in this case let see it was subtraction. So, 0 1 0 0 minus 1 0 0 1, 1 0 1 1, so 0 1 0 0 is positive 4 and 1 0 1 1 is negative 5 I want to subtract negative 5 from positive 4. So, the result is actually 9, 4 minus, minus 5 is 9 we know that it cannot be represented using 4 bits, let see how this is detected 0 1 0 0 and negative of 1 0 1 1, negative of 1 0 1 1 is plus rry subtraction of 1 0 1 1 is a same as addition of 0 1 0 1 you add these two now, let us look at what happens here.

So, 0 plus 1 is 1, 0 plus 0 is 0, 1 plus 1 is 0 the incoming carry is 1; however, the outgoing carry is 1 plus 0 plus 0 it is 1, the outgoing carry is 0. So, the incoming carry is 1 here, the outgoing carry is 0 which means there is a overflow in this, even though you get a result 1 0 0 1 if interpreted directly it will tell you minus 7. But if you looked at what is the incoming carry to the last stage and the outgoing carry out of the last stage these two are different, then there is a problem.

Typically what computers do is, if something like that happens the circuitry can detect this overflow and they will go and setup flag call the overflow flag. So, if you have done 8085 programming, 8086 or 8286 and so on they would you have talked you about this set of flags that are there in the processor. So, the addition of two numbers or subtraction of two numbers can result in the overflow flag being set, which means the result that you have is not correct there is something that has happen to because it is not in the range that is represented.

(Refer Slide Time: 26:18)



Adder/subtractor unit.

So, I want to end this lecture with this nice circuit for doing 2’s complement addition or subtraction. So, if I want to do x either add y or subtract y, remember x and y are both already 2’s complement numbers, they could be positive or negative I do not care, x is a 2’s complement number and y is another 2’s complement number, if I want to design a circuit that is either adder or sub tractor. So, I give a control from the outside which is add bar slash sub, add bar slash sub the interpretation is if add bar slash sub is 0 it means that you are adding if add bar slash sub this 1, it means that you are subtracting.

So, let us look at that this circuit is beautiful the reason why it is interesting is, this n bit adder is an unsigned adder, taking an unsigned adder if you want a 2’s complement adder all you have to do is, if I want to do x plus y then put x here, put y here and set add bar slash sub to 0. So, if I put 0 here what would happen is, you would have y n minus 1 XOR 0 that is y n minus itself, y 1 XOR 0 is y 1, y naught XOR 0 is y naught and so on.

So, we are giving the number x, we are giving the number y as it is, but with the carry in of 0, if you do that then the some that you get, where is actually a 2’s complement representation of x plus y. In this case you have to ignore the carry that is coming out, you look at s n minus 1 to s naught that will be a 2’s complement result of adding y to x, if I want to subtract y from x. So, again x and y could both be positive or negative, I take x in it is 2s, so x is already in it is 2’s complements representations.

So, we put that x here, we put y here and we subtract to subtract we put sub equals 1 add bar slash sub equals 1. So, if you put one on each of the inputs of these XOR gate, so this output will be 1 XOR y naught which is y naught bar, this is 1 XOR y 1 this is y 1 bar, so on up to 1 XOR y n minus 1 which is y n minus 1 bar, this is equivalent to complementing the bit vector y.

So, remember if I want to do 2's complement subtraction, so we do negation of the input and add to do the negation this negation itself is 2 steps in 2's complement, you have to flip all the bits and add 1 to it. So, this part of the circuit is flipping all the bits and remember for add bar slash sub if I put sub equals 1 the c naught will go as 1 this will take care of adding 1 to the results. So, essentially if sub equals 1 we will have x n minus 1 to x naught added to y n minus 1's complement to y naught complement and then another one will be added to it.

So, that will be the result that you will get and this number will again be a 2s complement number. So, the interpretation can be a direct 2s complementation number, so this is a very beautiful circuit and this is how it is internally implemented in processors. So, this brings me to the end of module 49 for this week and in the next module we go and look at how to design an adder itself.

So, in this picture for instance I said let us assume that there is a n bit adder that can do unsigned additions, it should suppose to take two numbers x and y and may be a carry bit that is coming in and give you a carry and a sum. So, assume that this adder is a black box here, in the next video I want to show how do design this black box. Once, you know that black box coming up with the 2's complement adder, sub tractor circuit is straight forward.

So, subtraction is not completely different all it required is a bit of XOR gates, if I want to n bit sub tractor all I have to do is put n more XOR gates that is all, it is as simple as that for 2's complements, 1's complement is slightly more tricky, 2's complement is what is used in most computers. So, I will leave it here for next video, we will look at unsigned addition in the next video.

Thank you and I will see you in a little while.