

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 - 48
Number Representation

Welcome to week 9 of this course, so we are at the last week of this course. So, we started with 8 week plan and we extended that to 9 weeks, to accommodate of few additional topics that were requested by several students. So, in this week we will look at Number Representation and we will look at arithmetic circuits. So, we have touched upon some of these things in several other weeks so far.

But in this week we will consolidate all of the things that we have seen so far. In fact, you have done some programming assignments already based on some of the knowledge that you have. In this video and in this week we will take all of these concepts and formalize all of the ideas. So, we will start with number representation, so the first number representation that we started within the beginning of the course is positional number system.

(Refer Slide Time: 01:18)

Positional Number System – a review


- Value of a number is determined by a *weighted sum* of its digits
- Weighting is implicit and is determined for each digit by the position of the digit in the number
- Let the number have n digits to the left of the radix point and m digits to the right of the radix point. D_i is the i^{th} digit, R is the *radix* or *base* of the number, V is the *value* of the number.

$$V = \sum_{i=-m}^{n-1} D_i R^i$$

[n digits] • [m digits]

Example : $12.34_{10} = 4 \times 10^{-2} + 3 \times 10^{-1} + 2 \times 10^0 + 1 \times 10^1$

Other common radices – binary (2), octal (8), hexadecimal/hex (16):
0-9, a, b, c, d, e, f.

 NPTEL

Number Representation 2

So, positional number system is one in which the position of the digit dictates a weight

and we added weights appropriately that gave us the value. So, in all the... So, in this video what we will see is, the notion of, if you are given a bit vector, a sequence of bits. So, you are given a collection of bits, we will see how to interpret them properly. In a positional number system, if I give you an n bit vector, then we treat each of these bits as though they indicate a weight and we have a simple formula which will convert the weights to the appropriate value.

So, for example, if I give you a base R and let us say D_i is the digit at position i , this could be in any base and let us say the bases R , then for and let us say there are numbers of this form. There is some n digit number before a decimal dot and there is m digit number after the decimal dot. So, these digits could be decimals, these could be hexadecimal, binary, octal it does not matter. Whatever it is, if you are given a number of this form, where it is n digits followed by m digits.

Then, the value that we attached to it or the interpretation that we have for this one is, for i equals minus m to n minus 1. So, minus m all the way up to n minus 1, we take $D_i R^i$, so that is the digit at position i multiply by the base to the raised i . So, this R^i is the weight that is assigned to the position. So, for example, we looked at this 12.34 base 10, so this has two digits in the front and two digits after decimal.

So, you start with this one, this is 4 into 10 power minus 2 plus 3 into 10 power minus 1 plus 2 into 10 power 0 plus 1 into 10 power 1 that is the equivalent value and that is the representation 12.34. So, if I give you some number like 1 1 1 dot 1 0 1 base 2, then you would be able to interpret that by putting an appropriate weights and so on, this is something that we already know. So, if you given a vector what happens is, this sequence of digits that you see as a vector, if you want interpret that as a value and if you think that it is a number, then we are interpreted by assigning appropriate weight and so on.

So, the positional number system is something that we have done, it is straight forward, we have seen this before. So, we will start with the so called representation of sign numbers. So, so far we have not discussed, how numbers both positive and negative are represented and this is the routine thing that we deal with in computers. So, computers should be able to store and manipulate negative numbers also, it is not that all the operations that we have are positive in nature.

For example, a bank balance could go negative, so things like temperature could go

negative and so on. So, we want to be able to do arithmetic operations like addition, subtraction, comparison, multiplication and so on, on negative numbers also, which essentially means first of all we need a representation for the negative numbers. So, as humans we look at a sign which is minus in front of a number and we say this is the negative number.

But, we need a representation which will also accommodate what are called sign numbers, sign numbers mean the numbers are... So, the sign numbers represent both positive as well as negative numbers. So, basically if you go and look at the number series, there are infinitely many numbers both positive and negative and however, a computer typically has a fixed storage. So, for example, if you go and look at most computers that are available nowadays, you will either see that they are 32 bit computers or 64 bit computers.

So, a 32 bit computer is one in which the internal representation allows you to have only 32 bits, which means you can represent 2^{32} distinct values. In this 2^{32} distinct values, you have to accommodate both positive and negative integers. Similarly, if I go and look at a floating point value, a floating point value is typically also 32 bits which means you can represent 2^{32} distinct floating point numbers.


So, basically it is possible that you are given some storage, so let us say 32 bit number and the number that you want to represent is much larger than that, which means this cannot be natively stored in your computer and manipulated on it. As long as number can be natively stored and manipulated, so there are circuits inside which will allow you to do that and you can exploit it. However, if a number goes outside the range of what can be natively represent in the computer, you need software techniques to deal with these.

So, what we will do first is we deal with, we look at four popular mechanisms by which we deal with sign numbers. So, sign numbers are those in which we want to represent negative numbers or negative integers to be more precise, 0 and positive integers. So, the floating point representation is slightly outside the scope of this course, we will look at only sign numbers in the videos that follow.

(Refer Slide Time: 06:45)

Representation of Signed Numbers

- How to represent positive and negative integers using 1's and 0's of the binary notation ?
- In mathematics, there are infinitely many positive and negative integers. However, in a practical hardware system only a fixed number of integers can be represented.
- In most modern computer systems, numbers are represented in 32 bits. If an arithmetic operation results in a number outside the range, an *overflow* occurs.
- There are 4 popular schemes for representing signed numbers.
 1. Sign Magnitude
 2. Ones Complement
 - Twos Complement
 - Excess-B or Biased Representation



Number Representation 3

So, there are four ways in which the sign numbers can be represented. They are sign magnitude representation, ones and twos complement representation and excess B or biased representation.

(Refer Slide Time: 06:58)

Sign Magnitude Representation


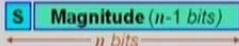
- Use **MSB** as a **sign bit** as follows,
 - MSB = 0 for positive integers
 - MSB = 1 for negative integers
- Other bits encode magnitude of integer.
- Range of representation with n bits is: $-2^{n-1} < X < 2^{n-1}$
- The range is symmetric around 0.
- There are two representations for 0, i.e., 0000 and 1000.
- Examples:**

Assume a 4-bit representation,

5 = 0101
-5 = 1101
3 = 0011
-7 = 1111

Convert the following sign magnitude numbers to a 6-bit representation.

0101 =
1010 =



Number Representation 4

Let start with the first one, sign magnitude representation. So, in a sign magnitude representation what we have is, if you are given a n bit vector, we interpret the left most or the most significant bit as the sign and the rest as magnitude. So, this gives us the name sign magnitude representation. So, n bit vector will be interpreted as a 1 bit that is

for sign and $n - 1$ bits, that is for magnitude.

So, the interpretation of this bit, the most significant bit is, if S is 0 then it is consider positive and if S is 1 it is consider negative. And let us look at the range that this can represent, if I give you a n bit vector, then one of the bits is already lost to the sign. But then you have $n - 1$ bits, for the $n - 1$ bits, this $n - 1$ bits can go from all the way from 0 to $2^{n - 1}$. So, with the sign magnitude representation, this has being 1 or 0 essentially doubles that range, we should able to represent numbers from $-2^{n - 1}$ all the way to $2^{n - 1}$.

And since we have this restriction, so if I give you a 4 bit vector unsigned representation, if I give you 4 bit vector, you can only go from 0 to 15, you cannot go to 16. So, that is why you have a strictly less than $2^{n - 1}$ here and X is strictly greater than $-2^{n - 1}$. So, if you are given only 4 bits for sign magnitude representation, you will not be able to represent 8 and minus 8, you can represent all the way from minus 7 to 7.

So, the range is symmetric around 0, what that means is, you have as many positive numbers as there are negative numbers and 0 itself actually has two representations. So, both 0 followed by all 0's and 1 followed by all 0's are the both stands for 0. So, this is actually positive 0, because you have sign as 0 and magnitude is 0 and this is negative 0. So, sign is 1 and magnitude is 0; however, technically there is no difference between plus 0 and minus 0 for various arithmetic operations that we are going to do. So, 0 gets to have two representations in this.

So, let us do a bunch of examples, so let us assume that we want to represent numbers using 4 bits. If you want to represent numbers using 4 bits, let us start with number 5. So, number 5 is actually 1 0 1, the Titus representation is 1 0 1, but if I want to store this in a 4 bit representation, if I tell you that you have to store it using 4 bits, then the foremost bit has to be a 0. So, let us take a look at this, 0 is interpreted as a positive number. So, the rest of the digits signify a positive some whatever integer is and the rest of the numbers is 1 0 1 which stands for 5.

So, this is positive 5, let us look at this number here minus 5, so for minus 5 you put a 1 followed by 5's binary of representation that is going 1 0 1 itself. So, now what is the representation for 3, this is plus 3, so plus 3 is... So, 1 1 is actually there are Titus binary

representation for 3 and we have given 4 bits. So, the very first bit must be a 0, because it is plus 3. So, the first bit must be a 0 the most significant bit, then the next most significant bit should also be a 0, because 3 has only 1 1, so it has only 2 bits. So, the representation for 3 should be 0 0 1 1.

Now finally, think about what should this number being, so it is 1 followed by 1 1 1. So, this means it is a negative number and the magnitude 1 1 1 is for 7, so this is for minus 7. So, I want you to go and do this as a little bit of an exercise, convert the following sign magnitude representation into a 6 bit representation. So, assume that these are both sign magnitude numbers, but they are represented using 4 bits, I would like you to go and think about, how to do this in 6 bits.

So, I am going to give you the answer, but I want you to think about it, you may want to pass the video and then come back and look at the answers. So, let us look at this 0 1 0 1, 0 1 0 1 is a positive number and the value is 5 and we want that to be represented in a 6 bit representation. So, in a 6 bit representation we would still start with the more significant bit to be 0, because it is positive and 1 0 1 instead of 3 bits we will need 5 bits, the 5 bits will be 0 0 1 0 1, so that will be the 6 bit representation for 0 1 0 1.

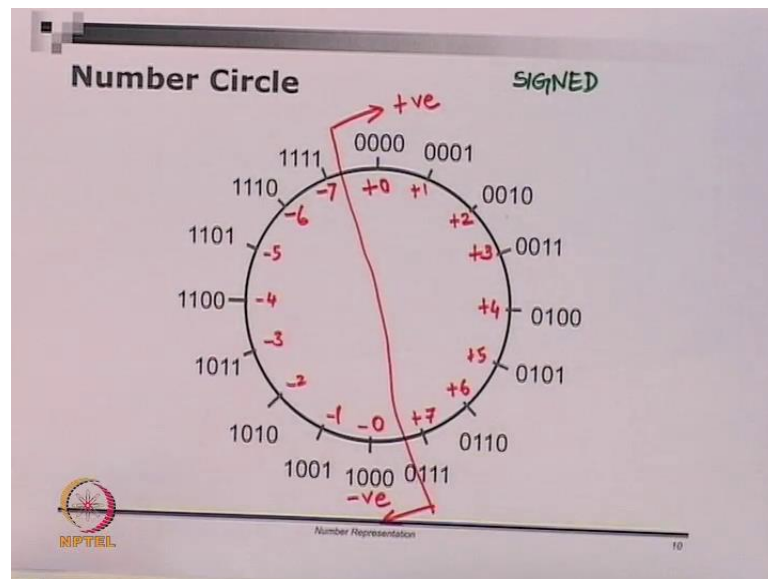
Let us look at this number below 1 0 1 0, 1 0 1 0 in a sign representation stands for negative 2, because 1 is for negative and 0 1 0 is 2. If you want this represented in 6 bits, then it should be 1 followed by three 0's followed by one 0. So, clearly if you have a representation which is in fewer bits, if you expand it to more bits, you should always be able to represent it. So, that is how most of the sign representations work. If you have a sign representation of a number in n bits, you should be able to extend it to some m bits which have greater than n, it should always be possible.

However, if you have a number represented with the larger number of bits, you may or may not be able to represent that using fewer bits. So, for example, if I give you number 9, writing 0 9 or 0 0 9 will extend the number of digits. So, I can write 9 as a 3 digit number 0 0 9, but if I give you a number 995, you cannot write that using 2 digits or 1 digit, you need at least 3 digits for it, so the same thing applies in binary numbers also.

So, what if you start with sign magnitude representation of n digits, you can always extend it to some n dash digits greater than n. So, this is sign magnitude representation, so to make this tick in your mind, I want you to look at this picture here, this is called a

number circle.

(Refer Slide Time: 13:37)



So, the number circle this pretty much looks like a clock, only that there are a few differences with respect to a clock. So, in this case the number circle is representing 4 bit numbers. So, let us start from here and let us move around the circle in the clockwise direction. If you go in the clockwise direction, you see that it start with 0 0 0 0 and then goes 0 0 0 1, 0 0 1 0, 0 0 1 1 and so on all the way up to 1 1 1 1. So, these are all the 4 bit values that you can have.

So, I am not talking about the interpretation of these numbers yet, it is just if I give you 4 bits, these are all the combinations that we have and this is arranged in some increasing order of the values. So, clearly if it is unsigned interpretation that I attached to it, then this position will be 0, this position will be 1, this position will be 2, this will be 3 and so on all the way up to 15.

So, a 4 bit unsigned value can go from 0 to 15, clearly you cannot represent anything above 15, you cannot represents 16 and above, you can also not represent anything below 0, you cannot represent any negative numbers. So, this is the deal with unsigned numbers. Now, I am going to show you how to deal with signed numbers. So, I have the same numbers circle here and in this number circle, it still going from 0 0 0 0, 0 0 0 1 and so on.

The clockwise it is looking like this odometer that I am keep talking about. It is like the odometer running, which runs from 0 0 0 0 all the way to 1 1 1 1 and then it will roll back to 0 0 0 0. So, if you keep going in the clockwise direction, it is always like incrementing the meter and if you go in the negative directions like decrementing the meter, so it is like the meter running back wise.

But, now I want a sign representation interpretation for this, so the vector that is given is 0 0 0 0, I want to see how to interpret it as sign representation. So, 0 0 0 0 is this indicates that 0 is a... So, the first digit is a 0 and the rest are all the magnitude, so this 0 is for a positive number and you have 0 0 0. Similarly, all the way here if you see, if you go from here all the way up till here the first bit is 0, which essentially means this half, this all positive numbers and this side is negative numbers.

So, let us see what these numbers are, so this is plus 0, this number would be plus 1, this vector will be interpreted as plus 2, this is plus 3, so on up to plus 7. If you start from here, this is minus 7, because it is this saying is negative number followed by 3 bits which stands for 7. This is minus 6, minus 5, minus 4, minus 3, minus 2, minus 1 and I already mentioned that we also have a negative 0 as well as a positive 0. So, this is a positive 0 and this is a negative 0.

So, let me finish this, this is the sign representation or sign magnitude representation or interpretation of the numbers and if you notice there are a few things. So, this is only for 4 digits, if I want to get this circular representation for 5 digits, you should draw circle which has more values in it. So, this has 16 values, you should be able to draw one with 32 of them and again the one half. So, if you draw a line from 0 0 0 0 to let us say slightly to the left of 0 0 0 0 to slightly to left of 0 1 1 1 or the equivalent 1, if you draw line everything to the right of that will be positive numbers, everything to the left of that will be negative numbers.

And you will see that 0 and minus 0 are in the exact opposite corners, so this is the sign magnitude interpretation of these 4 bit values. So, now let us go to the other interpretations. So, if I go and look at what is called ones complement interpretation.

(Refer Slide Time: 18:04)

Ones Complement Representation

- Positive integers are represented "as is". Negative integers are represented by performing bit-wise complement of the integer. E.g.,
 - 5 = 0101
 - -5 = 1010
- All **positive** integers have **MSB=0**; **negative** integers have **MSB=1**
- Range of representation with n bits is: $-2^{n-1} < X < 2^{n-1}$
- The range is symmetric around 0.
- There are two representations for 0, i.e., 0000 and 1111.
- **Examples:**

Assume a 5-bit representation,

01110 = 14
10111 = -8
15 = 01111
16 = not with 5 bits
-16 = not with 5 bits

Convert the following signed numbers to a 8-bit representation.

01011 = 00001011
10111 = 11110111

MPTEL

Number Representation

5

So, the ones complement interpretation is positive numbers are represented as it is, there is no change in positive numbers. However, negative numbers are represented by performing bit wise complement of the integer. So, let us look at this, if I want to get a 4 bit representation of the value plus 5, so the value plus 5 in binary is plus 0 1 0 1. So, if we can leave the plus sign, we can write it as 0 1 0 1.

However, if I want to represent minus 5 and if I want to represent minus 5 in ones complement form, then the way to do that is, you take whatever is the n bit representation of the positive number, negate or flip all the bits that is the ones complement representation. So, if I take positive number and negative number they will be flips of each other in ones complement representation. So, all the positive numbers will have MSB equals to 0 and all the negative numbers will have MSB equal to 1.

So, in according to this bullet point both sign magnitude representation and ones complement representation have this similarity. If MSB is 0, then it is positive all negative numbers have MSB equals 1. The range that you can represent is again very similar to the sign magnitude representation. So, it can represent from minus 2 power n minus 1 all the way to 2 power n minus 1.

So, this is again very similar to sign magnitude representation, the range is again symmetric around 0, which means you have as many positive numbers as there are negative numbers that you can represent with n and again as with signed magnitude

representation, there are two representations for 0, namely 0 0 0 0 which is for plus 0 and 1 1 1 1 if I want to interpret that in as a decimal value.

This 1 1 1 1 in ones complement interpretation is I take the first bit, the first bit tells me that it is a negative number. So, I will put the sign minus somewhere down and 1 1 1 1 I will complement all of the bits will be 0 0 0 0. I will read the binary interpretation of that 0 0 0 0, so it is negative 0. So, let us look at a few examples, so we are going to assume that we want a 5 bit representation.

So, we are going to go from 5 bit representation to decimal as well as go from decimal to 5 bit representation. So, let us see this number first, 0 followed by 1 1 1 0. So, this one tells me that, this 0 tells me that is a positive number, 1 1 1 0 is the magnitude in this case. So, the positive numbers are very similar to the sign magnitude form itself. So, this is 0 followed by 1 1 1 0 which means it is positive, 1 1 1 0 in binary is 14, so this is plus 14.

Let us look at this number here 1 0 triple 1, so 1 0 triple 1 stands for a negative number, because this is 1 and 0... So, if I take 1 0 triple 1 and I flip all the bits it is 0 1 0 0 0, so 0 1 0 0 0 if you interpret that as an unsigned binary number it is 8. So, my answer to this question is minus 8, so the first one is the plus 14, the second one is minus 8, now let us look at how to represent 15 positive number 15.

So, positive number 15 is 1 1 1 1 in unsigned binary and if I put a 0 in front of it, it is 0 1 1 1 and that is actually for a 5 bit representation of this number 15 in ones complement form. However, let us look at 16 and minus 16, so 16 in binary in unsigned representation, it is a positive number I should be able to represent that as unsigned number, unsigned number would be 1 followed by four 0's.

So, already 16 requires 5 bits to represent 16 you need 5 bits even in the unsigned form, you will not be able to represent that in the ones complement form or even in the sign interpretation. You cannot represent 16 using only 5 bits in the sign interpretation as well as in the ones complement interpretation. So, if you notice I want 16, so 16 is already 2^4 . So, 16 is 2^4 , I can represent only from minus 2^3 , so let us look at this.

So, we want a 5 bit a representation, so minus 2^5 minus 1 is minus 2^4 , 2

power 4 less than x less than $2 \text{ power } 4$. What; that means, is x can take the values from minus 16 to plus 16, but the ends excluded, we cannot represent the ends, we cannot represent minus 16 and 16. So, it is essentially minus 15 to 15 can be represented 16 and minus 16 cannot be. So, we cannot interpret this as, we cannot represent 16 using 5 bits.

However, if you are asked to represent using 6 bits that is possible for 16 you can represent that using 6 bits. So, let us look at some other conversions, in this case we are taking a number which is smaller bits and we want to represent that in a larger representation. So, 0 1 0 1 1, so this is a positive number by the way 1 0 1 1 stands for number 11, if you want to represent that in ones complement form using 8 bits, then it is just take the 0 and append as many 0's as you want before that till. So, in front of this till you get this value here.

So, this is 5 bits if I append three 0's here, the front then it is 0 0 0 followed by 0 1 0 1 1 that is the representation for this one. Now, let us go and look at this number 1 0 triple 1, 1 0 triple 1 I know that is a negative number, because it is starting with a 1. So, what is the value for this 1 0 triple 1 is to find out the value we no down the sign that is negative it is minus 1 0 triple 1 is if I flip all the bits it is 0 1 0 0 0 which is 8.

So, this is the binary equivalent or this is the ones complement into representation of negative 8 and if you want to convert it into a 8 bit representation from this 5 bit representation, all you have to do is add as many once before this. So, let us take this one, so 0 1 0 1 1 appear as it is we are added three 0s in the front and similarly 1 0 triple 1 is as it is we have added three 1's at different.

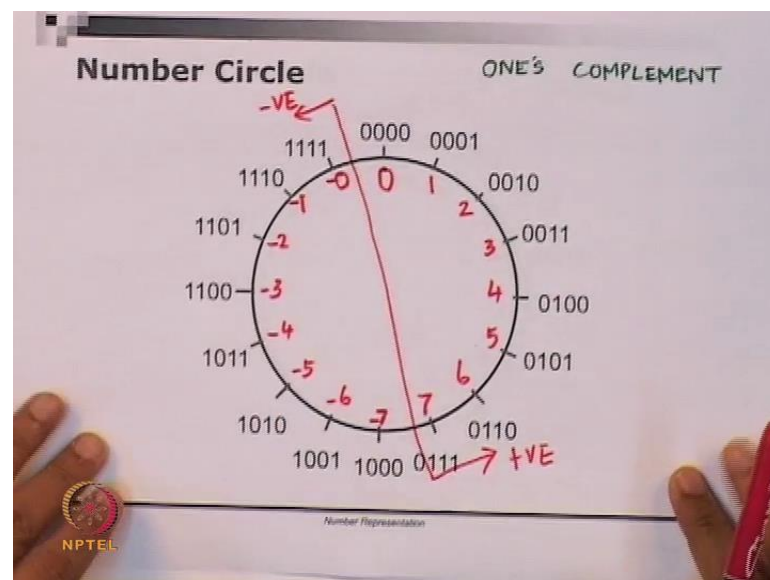
So, unlike the unsigned representation, remember these 1's do not actually hold any weight that is the key difference, if I tell you to go and interpret that in the unsigned form. So, this would be $1 \text{ into } 2 \text{ power } 7$, this would be $1 \text{ into } 2 \text{ power } 6$ and so on, if I ask you to interpret this 8 bit value in the sign magnitude form this one does not carry any weight. But this will start carrying weight from $2 \text{ power } 6$, $2 \text{ power } 5$, $2 \text{ power } 4$, $2 \text{ power } 3$ and so on; however, in ones complement the weights are not attached to the positions.

So, ones complement representation as well as twos complement, which will see later or not positional representation techniques. So, they have something which is partly position based and partly it is not. So, we would not call it a position representation

system, so in this case if I want to go and take this number and interpreted as a decimal value it is a same principle, you it is starting with a 1. So, it is a negative number flip all the bits, so it will be 0 0 0 0 followed by 1 0 0 0. So, 0 0 0 0, 1 0 0 0 8, so that is equivalent to minus 8 there.

So, if I want to extend number into more digits, more bits then take the most significant bit and append as many bits as you want to the left. So, that will be a valid ones complement representation. So, let us look at the number circle representation for ones complement. So, as before I have the...

(Refer Slide Time: 26:55)



So, I have 0 0 0 0, 0 0 0 1 and so on and it is an number circle, so if I want to get the interpretation this is number 0. And all the positive numbers are on the right side, if I go from 0, 0, 1, 2, 3, 4, 5, 6 and 7. So, at this point if you notice it is starting to become 1 as the first bit, so clearly these are not. So, this side is not positive numbers, they are all supposed to be for negative numbers.

Now, let us see what these numbers are this 1 1 1 1 is actually stands for minus 0, this 1 stands for minus 1, this for minus 2, minus 3, minus 4, 5, 6 and 7. So, these are the numbers, these are the values if these 4 bits or interpreted as ones complement. So, it is all about interpretation, we have 4 bits we want to interpret that in ones complement and that is how it is works. So, the way you can think about this is, if you go in the positive direction from 0 in this is clock clockwise direction.

It goes from 0, 1, 2, 3, 4, 5, 6, 7 and it stops at 7, because in this example n equals 4 and we cannot represent $2^4 - 1$ which is 2^3 or 8 we cannot represent positive 8 we also cannot represent negative 8 and one thing you can notice is minus 0 and plus 0 are actually neighbors of each other in the representation and if I go in the... So, again even in this case if I go in from here till here in the clockwise direction, the numbers are increasing.

Similarly, from minus 7 if I go in the clockwise direction, the numbers are increasing all the way to minus 0, this still increasing minus 7 to minus 6 means you are increasing minus 7 to minus 6. So, from minus 7 if you go all the way to minus 0 it is still increasing. So, there is a break here when we go from 1 1 1 1 to 0 0 0 0, similarly there is a break when we go from 0 1 1 1 to 1 triple 0.

So, even in this representation if I draw line here all the numbers to the right side are positive and all the numbers to the left side are negative and as in sign representation we have negative 0 and positive 0 both. So, there is this interpretation call the ones complement interpretation. Finally, let us look at this other representation call the twos complement representation, in this again the positive numbers are represented as it is negative numbers are calculated by subtracting the magnitude of the negative number from 0 and you borrow from imaginary position.

(Refer Slide Time: 29:41)

Twos Complement Representation

- Positive integers are represented "as is". Negative integers are formed by subtracting *magnitude* of negative integer from 0; borrowing from imaginary position to the left of MSB.
(Shortcut: bit-wise complement of the integer and add 1).
- Example: 5 = 0101 -5 = 0000 - 0101 = 1011
- All **positive** integers have **MSB=0**; **negative** integers have **MSB=1**
- Range of representation with n bits is: $-2^{n-1} \leq X < 2^{n-1}$
- The range is *asymmetric* around 0.
- There is only *one representation* for 0, i.e., 0000.
- **Examples:**

Assume a 5-bit representation,

01110 = 14

10111 = -9

15 = 01111

16 = *not with 5 bits*

-16 = 10000

Convert the following signed numbers to a 8-bit representation.

01011 =

10111 =

Number Representation
6

So, as an example what we do is, so there is a short cut way of getting a twos

complement representation, you take the number if it is a positive number you write it as it is, if the number is negative. So, of the first thing you do is complement, so you represent that number as though it is a positive number, if I want to 5, so 5 is which as 0 1 0 1 we represent as it is, if you want minus 5 we take 0 1 0 1 which is the positive 5 flip all the bits they will give you 1 0 1 0 and add 1 to it that be 1 0 1 1.

So, that is the twos complement representation of negative 5, so 1 0 1 1 in twos complement interpretation would be minus 5, here also positive integers are all 0, negative integers have MSB equal to 1. So, positive integers have MSB equals 0, so the range; however, is an interesting thing. So, there is a small change with respective ones complement, in terms of the positive numbers you cannot represent more numbers then what is ones complement is capable of.

So, if I tell you, you have to represent something with n bits then you cannot represent 2^n power n minus 1, you can represent a number which is 1 less than 2^n power n minus 1. However in the negative numbers you can represent one more number, so that is why you have a less then equal to here so; that means, the ranges asymmetric around 0 and it also means that the 0 is does not have a 2 representations.

So, if I have n bits I can represent 2^n power n distinct values, the 2^n power n distinct values in twos complement are all the way from 0 to 2^n power n minus 1 and all the way from minus 1 to 2^n power n minus 1 inclusive. So, in the positive directions it is 0 to 2^n power n minus 1 minus 1 and the negative direction is minus 1 to minus 2^n power n minus 1. So, let see some examples, we will take the same things that we did before. So, 0 followed by 1 1 1 0 if I want to interpret that if this is a twos complement interpretation, in this a positive number 1 1 1 0 is 14, so that is 14 itself.

If I want to interpret this, this 1 0 triple 1, so I know it is a negative number, if I want the decimal equivalent of it I first of all flip all the bits. So, it is 0 1 triple 0 and then I add 1 to it, so it 0 1 triple 0 I add 1 to it is 0 1 0 0 1 0 1 0 0 1 is 9. However, since we started with the number MSB equal to 1 it is not 9, it is minus 9 then let us look at 15 how to interpret how to represent 15.

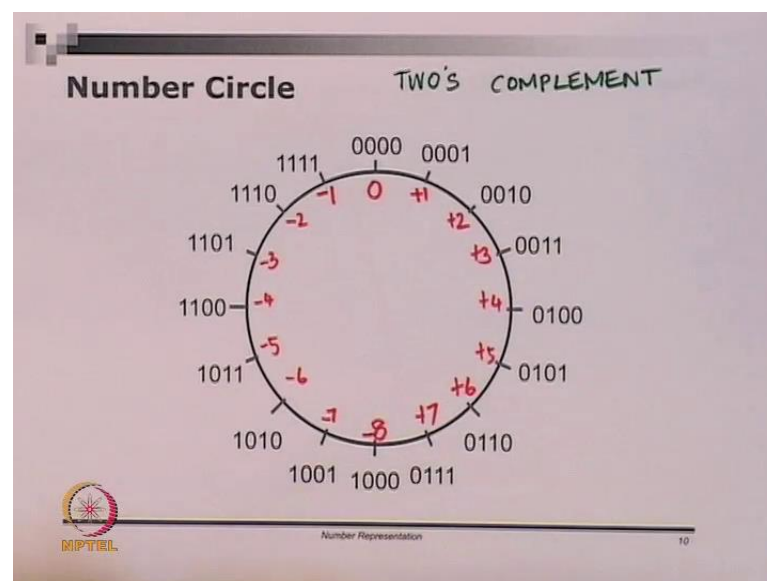
So, 15 is 1 1 1 1 in binary that is only 4 bits, so I am asking it to be represented in 5 bits all you have to do is put a prefix of 0. So, 0 are followed by four 1s is plus 15, plus 16 in the binary representation unsigned form is 1 followed by four 0's. However 1 followed

by four 0s if I represent that in 5 bits, then it is now going to be interpreted as a negative number. So, 16 cannot be represented in 5 bits, if there is a two's complement interpretation because we know that 2^4 power...

So, if it is 5 bits, so $2^5 - 1$ is 2^4 is 16, so x has been less than 16; however, minus 16 is possible. So, for minus 16 it is 1 followed by four 0's that is 16 flip all the bits that is 0 followed by 1 1 1 1 you add 1 to it, it is again 1 followed by four 0's that is the representation for minus 16 in 5 bits. So, for minus 16 it is actually in this looks like it is positive 16, 1 followed by four 0s is positive 16 the unsigned interpretation, but it is minus 16 in the two's complement interpretation.

As before you can take a number with the lesser number of digits and extend it to more digits. So, 0 1 0 1 1 it is 5 digits if I want to represent that as a 8 bit number I can prefix 0 to it three 0's. Similarly, if I want to extend the negative number into more bits I can add more 1's to the front. So, this is just like what we had for ones complement interpretation, so going from a smaller number of bits to higher number of bits usually requires only sign extension you take the more significant bit and copy to all the positions which you have added that is enough for two's complement interpretation, the last...

(Refer Slide Time: 34:51)

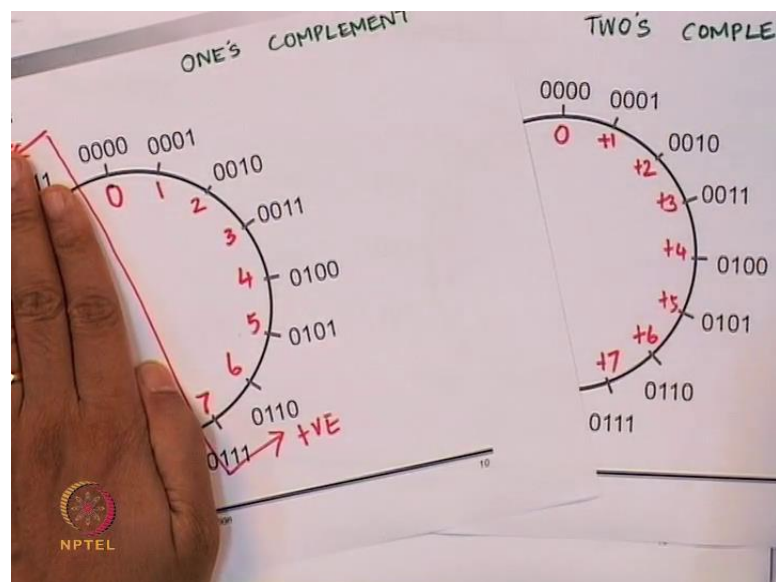


So, now let us look at the number circle for two's complement, for two's complement we start with 0 here and it goes as. So, in fact there is no positive 0, negative 0 it is only 1 0

here then plus 2 plus 3 plus 4 plus 5 plus 6 and plus 7. So, here also if I draw an imaginary line here everything to the right of that is positive or 0 everything to the left of that is negative and there are negative numbers start here, it goes in this order minus 1 minus 2 minus 3, 4 minus 5 minus 6 minus 7.

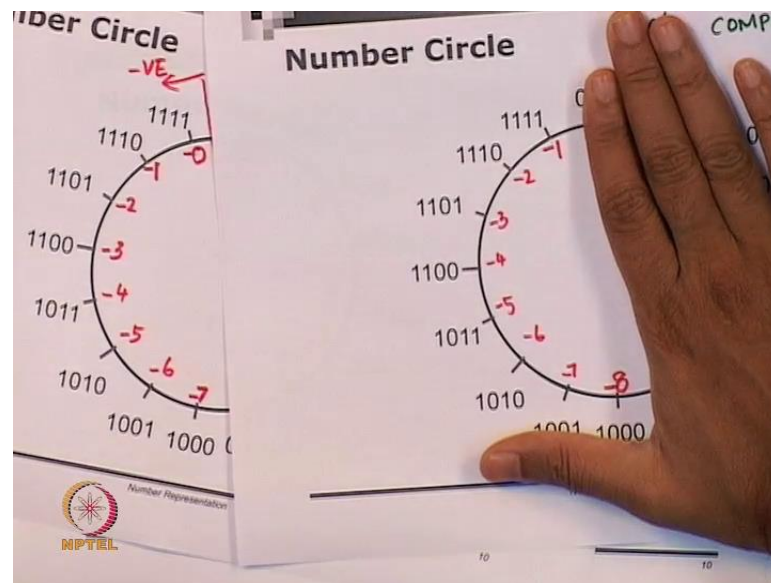
However, this 1 triple 0 is minus 8, so we will have 0 and minus 8 opposite each other, if it is a 5 digit representation 0 and minus 16 will be opposite to each other and if it is a 6 digit will be 0 and minus 32 and so on. So, this is the number circle for twos complement. So, I know that many text books do not carry the number circle, but this is a very useful way to remember how interpretations are done in ones complement verses twos complement and so on.

(Refer Slide Time: 35:55)



So, one thing I want to do is put the ones complement and twos complement side by side. So, this is the picture for ones complement and this is the picture for twos complement. So, let us start with just the positive half's first, we can see that... So, let us see 0 to 7, so if say 0 to 7, 0 to 7 has a same interpretation for all the positive numbers in both ones complement and in twos complement, so the representation does not change.

(Refer Slide Time: 36:21)



However, if I take the negative numbers, so these are all the negative numbers in twos complement, these are all the negative numbers in ones complement. So, if you see these numbers verses these numbers, so the right side one is twos complement if you see this, then there is one half. So, where we have minus 0 we have minus 1, where we have minus 1 we have minus 2, where we have minus 2 we have minus 3 and so on.

You can see that if the numbers are negative, then take ones complement and if you remove one from yet that gives you the twos complement interpretation. So, this is trick to remember and this is a common thing that people do as a trick.

(Refer Slide Time: 37:07)

Excess-B (or Biased) Representation

- Integer representations are *biased* by B .
- A signed integer X is represented by the binary number $X+B$
- Range of representation with n bits is: $-B \leq X < 2^n - B$
- Usually, $B=2^{n-1} \Rightarrow -2^{n-1} \leq X < 2^{n-1}$
- There is only *one* representation for 0, i.e., binary representation for bias B .
- Examples:**

Assume a 5-bit representation and $B = 2^4$.

10001	=	$17 - 2^4$	=	1
01100	=	$12 - 2^4$	=	-4
00000	=	$0 - 2^4$	=	-16
0	=	$0 + 2^4$	=	16
11111	=	$15 + 2^4$	=	31

NPTEL

Number Representation

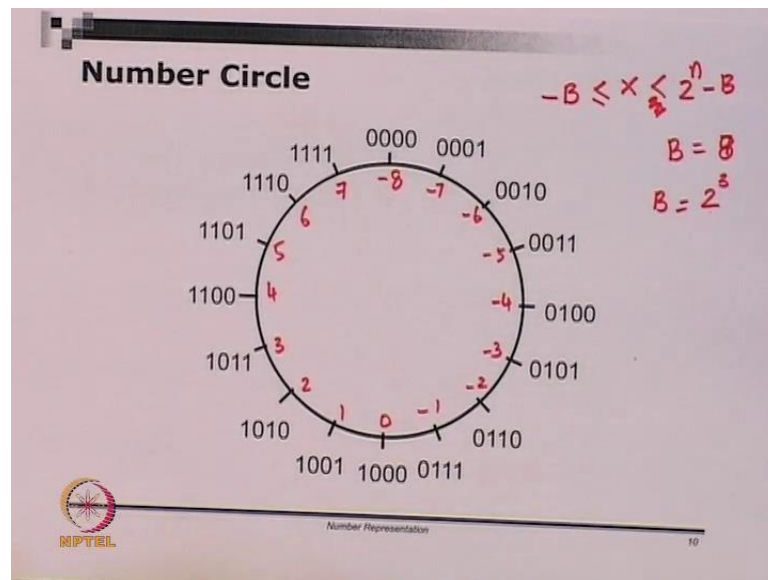
7

So, finally, we will look at excess B representation or biased representation, in this representation integer representations are biased by B. So, a sign number X if it has to be a express using excess B you add X to B that is why it is called excess B. So, B excess you add B to it and if you do that then the range of representation with n bits goes from minus B through 0 all the way up to 1 less than 2 power n minus B.

Because, you are representing with excess B, you cannot represent 2 power n minus B plus 1, you can represent only up to 2 power n minus B minus 1. So, usually B itself is big as 2 power n minus 1 we do not do excess 5, excess 6 and so on we do excess 3, excess 7, excess 15 and whatnot, usually B is picked in such a way, so that the complete range of bits can be interpreted.

So, if B is 2 power n minus 1 then we can represent all the way from minus 2 power n minus 1 to 2 power n minus 1 minus 1. So, in terms of the interpretation this is very similar to twos complement. Because, it is starts from minus 2 power n minus 1 it goes all the way up to 2 power n minus 1 minus 1; however, and 0 has only one representation. So, both sign magnitude and ones complement has two representations for 0, whereas excess B and twos complement has only one representation for a 0. So, I do not want to draw this in the number circle, but I believe this is something you can look at, so this is not a very hard thing.

(Refer Slide Time: 38:56)

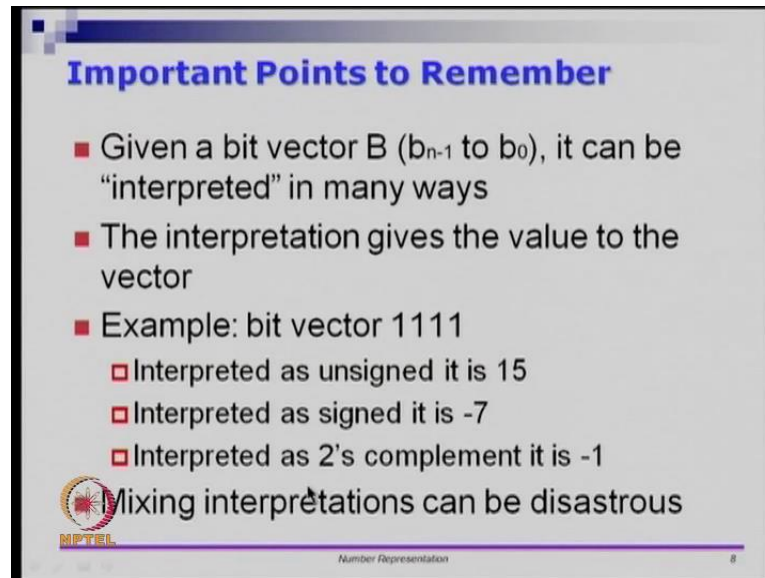


So, I am going to show this in the number circle, so we know that in the number circle representation we can go from minus B less than or equal to X less than 2 power n minus B. So, if I say that B is 7 or excess 7 then we should be able to go from minus 2 power 3. So, B is represented using 3 bits, so we can go from minus 8 minus 7, 6 minus 5, so in this case excess 8 representation. So, minus 5 minus 4, 3, 2, 1 this is 0, 1, 2, 3, 4, 5, 6, 7.

So, in this case B itself is picked as 2 power 3, so this is excess 8 interpretation, so this is the very useful thing. So, this is the number circle for this interpretation, so what you will see is I will later upload a set of PDF files which has a very nice pictorial view of whatever the different interpretations and where are they used and while. So, it is a nice set of pictures that is there at OCW's website I will post that along with the slides for this one.


So, let us look at some examples now for a 5 bit representation and B equals 2 power 4, we have 1 triple 0 1. So, if I interpret that is a 16 plus 1 which is 17; however, B is 2 power 4, so I have to subtract 16 from yet. So, this is 17 minus 16 which is 1, then 0 1 1 0 0 is if I interpreted directly it is number 12, but it excess 16. So, 12 minus 16 is minus 4, so this is the interpretation for minus 4, then 0 followed by all 0's it is already we have added 16 to it. So, I have the subtract 16 from it, so it will be minus 16 and so on and I suggest that you can go and do the representations for 0 and 15 yourself.

(Refer Slide Time: 41:04)



Important Points to Remember

- Given a bit vector B (b_{n-1} to b_0), it can be “interpreted” in many ways
- The interpretation gives the value to the vector
- Example: bit vector 1111
 - Interpreted as unsigned it is 15
 - Interpreted as signed it is -7
 - Interpreted as 2's complement it is -1

 Mixing interpretations can be disastrous

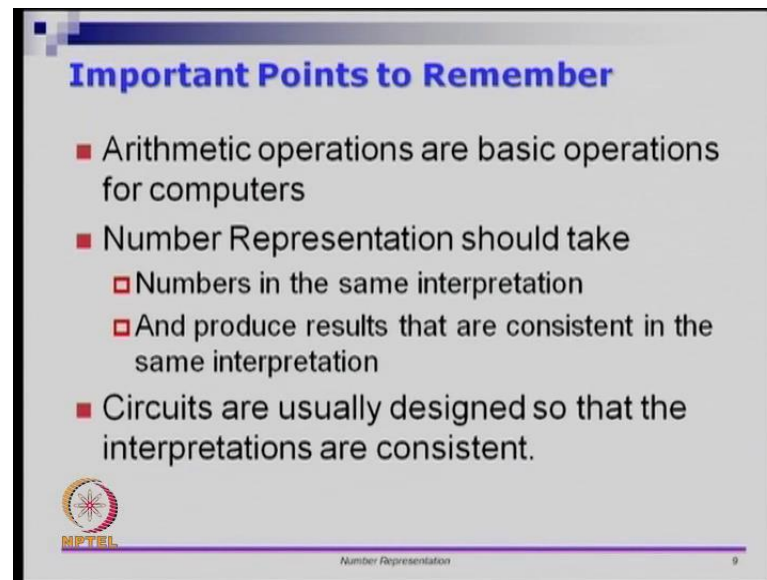
Number Representation 8

So, there are a few important point to remember given a bit vector B which goes from b_{n-1} to b_0 it can be interpreted in many ways, it is all up to the interpretation. So, given a n bit vector and if somebody ask you what is the value for this, the first question you should ask is what is the interpretation that you have to follow. The interpretation will give you a value, but if you are going from one interpretation to gather it may actually give different values.

So, let us look at this bit vector 1 1 1 1, if I ask you what is the value for 1 1 1 1, you should stop me and ask, what is the interpretation you should use. So, 1 1 1 1 in the unsigned representation is number 15, we have been using that all along, but the moment you use a sign interpretation, then this one is in the sign and 1 1 1 1 is the magnitude, so that is 7, so this is negative 7.


If you interpret the same thing using twos complement then it is minus 1, so we should be very careful not to mix the interpretations, mixing one interpretation the other can be very disastrous.

(Refer Slide Time: 42:14)



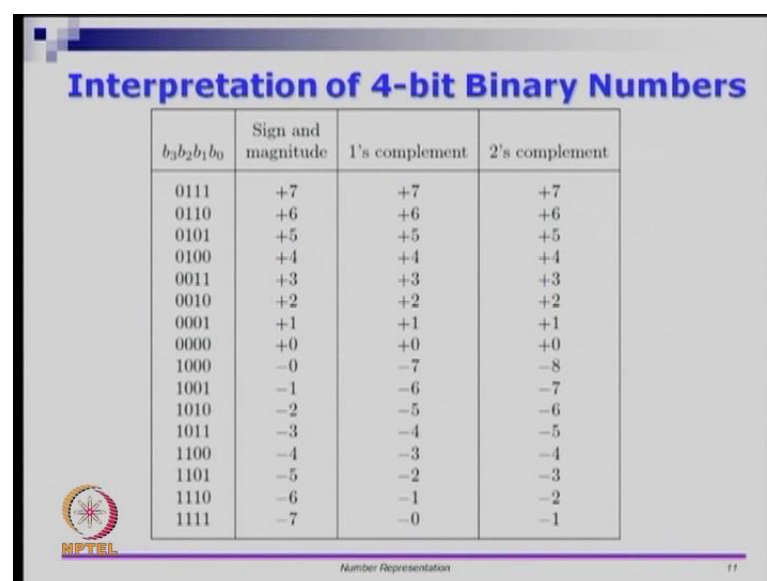
Important Points to Remember

- Arithmetic operations are basic operations for computers
- Number Representation should take
 - Numbers in the same interpretation
 - And produce results that are consistent in the same interpretation
- Circuits are usually designed so that the interpretations are consistent.

 Number Representation 9


So, there is something else that you should remember, arithmetic operations are basic for computers and the number representation should be able to take two numbers in the same representation and produce results that are consistent in the same representation. So, if I have a circuit that is supposed to be adding numbers, you cannot ask for a number you cannot ask for an adder which will take one signed number and another unsigned number and add them and so on, it does not make sense. Typically, you take two numbers which are interpreted with the same form and the result that you should get when interpreted in the same form the result should all be consistent that is how we will design circuits.

(Refer Slide Time: 42:55)



Interpretation of 4-bit Binary Numbers

$b_3b_2b_1b_0$	Sign and magnitude	1's complement	2's complement
0111	+7	+7	+7
0110	+6	+6	+6
0101	+5	+5	+5
0100	+4	+4	+4
0011	+3	+3	+3
0010	+2	+2	+2
0001	+1	+1	+1
0000	+0	+0	+0
1000	-0	-7	-8
1001	-1	-6	-7
1010	-2	-5	-6
1011	-3	-4	-5
1100	-4	-3	-4
1101	-5	-2	-3
1110	-6	-1	-2
1111	-7	-0	-1

 Number Representation 11

So, I will leave you with this table interpretation of 4 bit binary numbers, this is something that is very handy, if you want to later go and do some arithmetic I suggest that you go and keep this with you and work with us. So, this will come and quite handy later, so I am just leaving it for completion shake. So, this brings me to the end of lecture 1 for this week. So, in the next module what we will look at is, we will look how to do additions and subtractions in ones complement and twos complement representations.

So, thank you and I will see you in a little while.