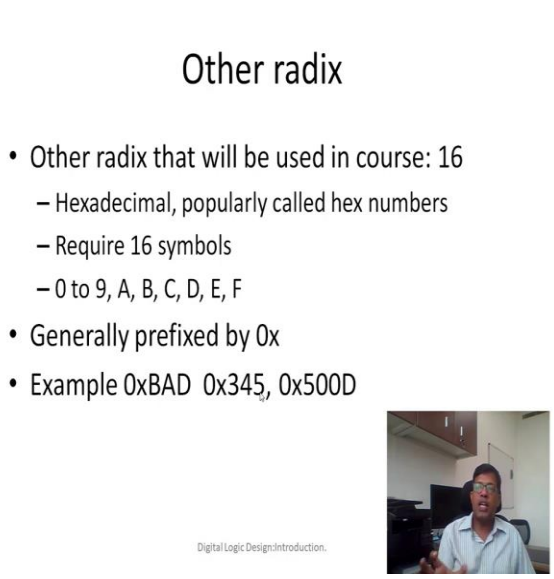
Digital System Design
Professor. Neeraj Goel
Department of Computer Science Engineering
Indian Institute of Technology Ropar
Lecture 4
Binary Number System 2

(Refer Slide Time: 00:18)



So, the other base which we will use is 16. So, 16 is also called hexadecimal number and in short we also call it hex numbers. So, those who are doing programming from initial ages, so you would see these hex numbers are quite popular, so many of these digits, they are many of these like programs are written in form of hex numbers and we are told about this in different aspects. So, hexadecimal means there are 16 symbols, because base is 16 we would require 16 different symbols.

How to get this 16 different symbols? First 10 symbols we have borrowed from the decimal number 0 1 2 4 5 6 7 8 9, but after that it would be confusing, so what has been done we have borrowed 6 symbols from the alphabets, A, B, C, D, E, F these are the 6 symbols we have borrowed from the, from our alphabets. So, whenever A is written, A represents 10, B represents 11, C represents 12, D represents 13 and E represents 14 and F represents 15.

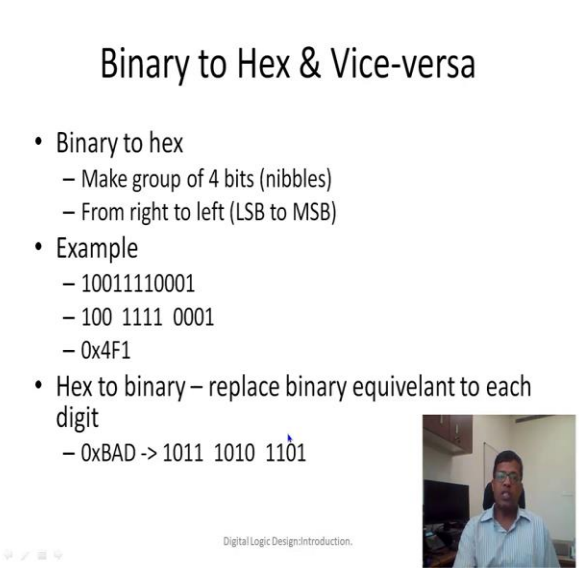
So, it does not matter whether this A, B, C, D, E, F is written in CAPS or in small letters, but the important thing is that all the, these many of these symbols would be presents. So, usually we need not to explicitly say base 16, whenever a number is prefixed by 0x, this again x could be

capital, x could be small, if a number is prefixed by 0x then we can assume that this particular number is a hexadecimal number.

The examples are 0 cross BAD, you can also read as BAD and 0 cross 345, so you see in 345 there is no mention of A, B, C, D, E, F, but because 0 cross x is written, so we will assume directly that this number is a hex number. So, there could be only this A, B, C, D, E, F present then we can directly find out that this number is a hexadecimal number. But if there is a possibility that none of this letters are there but only 0 to 9 numerals are there, but the number could be hexadecimal if it is mentioned that 0x is there.

Now let us do quickly that why do we use hexadecimal numbers, so essentially the major advantage of these hexadecimal numbers is that they are very good representation of binary number. So, let us say, my binary number has 32 bits, so writing these 32 bits would be confusing because we always have to keep a watch on what is the index of a particular bit, rather than that if I convert the same 32 bit number, 32 bit binary number into hexadecimal, total number of digits I will get is 8, so these 8 are easy to manage, easy to understand and they have one to one correspondence between following that binary number.

(Refer Slide Time: 04:03)



So, if we have to convert a binary number to hexadecimal number, so first thing we do is, we will make a group of 4 bits that is called nibbles and these nibbles we have to create from right to left, in other words from least significant bit to most significant bit. Let us take this example, so

this is a number 10011110001, so I need to convert this binary number into a hexadecimal number, so first thing I have to do is, I have to create a group of 4 bits starting from right to left.

So, if I am starting from right to left, then first 4 groups, first 4 bit or first nibble will be 0001, so that means this group and then the next nibble would be 1111, so I have kept a space in between so this is the next nibble. The third nibble is 100, so you have only 3 bits here not 4 bits, so whatever are the remaining bits that will be there in the last nibble.

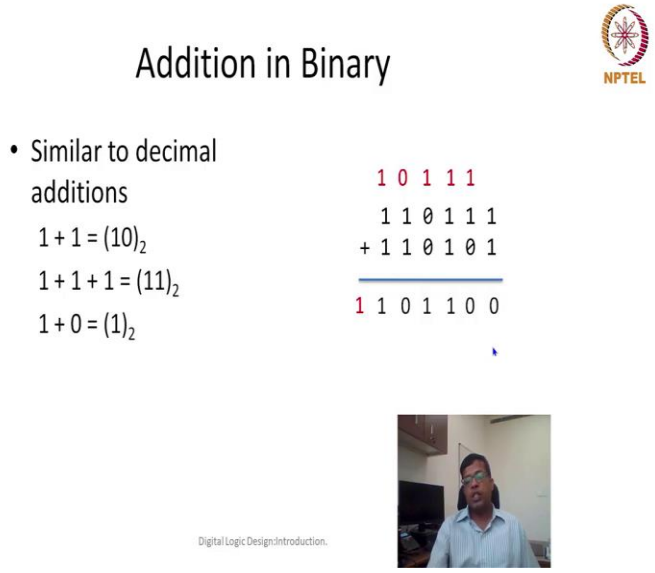
And after this nibbles are done or group of 4 bits are done what you will do is, you will look like a lookup table, so essentially you know that 0001 means 1 in hex and 1111 is essentially 15 and 15 in hex is F, so we write F here and 100 is 4, so we write 4 here, so 0x4F1 is the hex representation of this binary number. So, in whenever we want to convert any binary number to hex number, we can simply create some kind of a lookup table in our mind, where we know what is the binary representation of all 0 to F, 0 to F symbols of my hexadecimal numbers and from those symbols if I know the binary I can have a direct translation.

Similarly, if hex numbers are given and we need to convert to binary, so we can replace each hex number with the equivalent binary number. So, equivalent binary number is again kind of a in mind lookup table, so for example here we have BAD, B means 11, so for 11 whatever is the binary number I will write 1011.

Similarly, for A I know it is 10 in decimal, so and for 10 binary equivalent is 1010. Similarly, for D it is 13, so 1101, so similarly any hex number can be converted quickly into binary numbers and that is the precise reason why hex is a equivalent representation of binary.

So whenever people write binary, so for example 32 bit numbers, so it would require 32 bits and we can get easily, we can easily get confused between 1s and 0s, how many 1s are there. So, that is why in most of the textbooks as well as most of the documents wherever we design digital systems and we write things in hex because there is a one to one and a quick mapping with from hex to binary. So, this is how we understand hexadecimal decimal numbers.

(Refer Slide Time: 07:53)



Now before going to the next topic, so let us also understand that we have understood the binary representation, hex representation and we are seeing all of these things in context of decimal numbers also. Let us say if we need to do some arithmetic operations like addition in binary or hexadecimal. So, how do we do that? It is very similar to decimal addition. So, the only thing is only difference is the symbols which we use is only 1 and 0.

So for example, I want to add 1 plus 1, 1 is binary because it is 0 and 1 symbol, so 1 plus 1 is 2 but I need to write 2 in binary that means 1 0. So, if I need to add three numbers, 1 plus 1 plus 1 then also equivalent of 3 in binary is 1, 1 so we can write it like this. 1 plus 0 is 1 of course, so I have to write a multiple bit number, how do we do that? Let us take it like an example, the way we do it in decimal in the similar way we do things in binary.

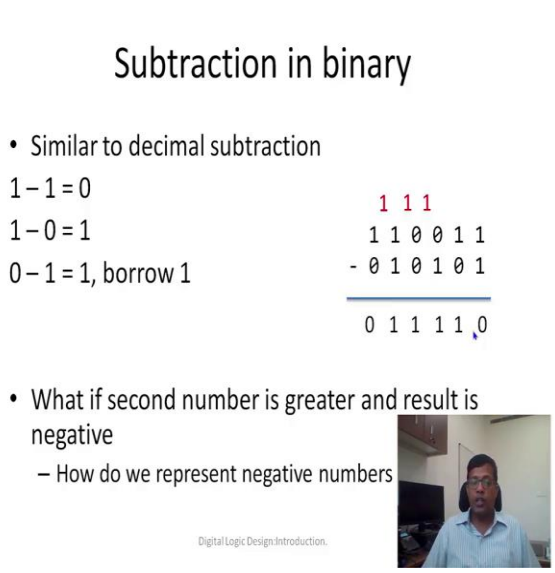
So, let us take this example, let us say we have these two 6 digit numbers, 6 bit number or 6 digit number, now we are adding both of them and just, whenever we are adding just keep in mind how do you do things in decimal numbers. So, first how do you do? You first take the least significant digit or here it would be least significant bit, the least significant bit is 1 and 1. Now we need to add them 1 plus 1 is 10, so we will keep 0 here and 1 would be carry.

So, now this time we need to add 3 numbers, carry 1 plus this 1 plus 0, 1 plus 1 plus 0 will again be 10, so that means I will keep 0 here and 1 would the carry. Now in this case there are 3 numbers to be added 1 plus 1 plus 1, so the result would be 11, so I will keep 1 here and will

keep a carry here. So, now this is 1 plus 0 plus 0, so I can write 1 and carry this time is 0. Now 0 plus 1 plus 1, the addition is 10, so that means 0 I will keep here and 1 would be the carry.

Now I can add again 1 plus 1 plus 1, so output would be, sum would be 1 and there would be a carry which is 1. So, essentially the answer for adding would be like this and you know the process is very similar to the way we do addition in our decimal numbers. The next question we can ask is that can we do a subtraction?

(Refer Slide Time: 11:02)



Possibly the subtraction is also can be done in a very similar way the way we do it in binary. So we can say 1 minus 1 is 0, 1 minus 0 is 1, but if I need to subtract 0 minus 1, so that means I need to take a borrow 1. So I take borrow it will become 10 and if 10 I subtract 1 from 10 then it will be output is 1, but we need to remember that we have taken borrow of 1.

So, the way we do subtraction in our decimal numbers, the similar way we can do subtraction in binary also, the only thing is we have to understand where we are taking borrow and what is the implication of taking borrow. So, borrow we need to return, so either we have to always say that this is the, by the end we will have this borrow. So, let us understand this again with a very similar example, so let us say we have again some 6 bit number which we need to subtract.

So if I want to subtract these two numbers, how do we start? So, again I will start with least significant bit, this is 1 1 here, so I want to subtract this 1 from this 1, the output would be 0 as we mentioned here. So, output is 0, there is no borrow, no carry so we have not written anything over here. So again 1 minus 0 is 1, again need not to borrow. So, we can write it like this, but here 0 I want to subtract 1 from 0, because I want to subtract 1 from 0 I need to take a borrow.

After taking borrow it will become 10 and from 10 I will subtract 1. So this borrow was already taken from the next number and from 10 I have subtracted 1, so the output is 1. Now what would happen here, I want to subtract 0 from 0, what about this borrow? So, essentially what I want to do is I want to subtract 0, I want to subtract 0 plus 1 from this 0, so again I need to take a borrow and from that borrow I would subtract this 0 and this 1, so 1 was already taken out from here, so that is why I will subtract, from 10 I will subtract this 1, again the output is 1.

So, I will repeat because from here 1 was already borrowed out, so that means I need to subtract 1 from this 0, so 1 I need to take borrow of 1, so it will become 10, so from this 10 I am subtracting this borrow, that is why from 10 it will become 1. Again let us see what would happen in the next case.

In the next case, we have 1 here and 1, now we again need to subtract, we again need to borrow 1 so that we can subtract this 1 and this 1, so by taking borrow it will become 1 1 that means 3, from 3 I will subtract this 1 and this 1, so that means from 3 I am subtracting 2 the output is 1 and what is the remaining thing?

Now here 1, so in the next bit, I need to subtract 1 0, so 1 minus 0 minus 1, so the output is 0, there is no borrow required further. Let us say there is a borrow in the last bit also, that means this whole number is my subtraction, whatever number I wanted to subtract that was larger and that is why in the end we are left with the borrow. So, this is how we do subtraction in binary.

So, let us say we are left with a borrow here, so which means the number which I was subtracting was a larger number, which also means my output is a negative number, if borrow is 1, so that means my output is a negative number, if output is a negative number the next question is how do I represent this negative number?

So, representing negative number in binary itself is a big question, so we will try to take up this question in our next lecture and by then thank you very much and enjoy doing tutorial questions and some practice questions which would be shared along with the lecture. Thank you.