

Digital System Design
Professor. Neeraj Goel
Department of Computer Science Engineering
Indian Institute of Technology Ropar
Lecture 5
Negative Number Representation 1

Hello everybody, in our last lecture we had a question that how we can represent negative numbers. So, today we will try to answer this question that how negative numbers can be represented and also we will try to answer the second question that if negative numbers can be represented then how addition, subtraction or other binary calculations can be done.

(Refer Slide Time: 00:48)

How to represent negative numbers?



- How to we do it decimal numbers
 - Negative sign
- Can we use negative sign in binary
 - Computer/digital world only understand 0 and 1
- How about reserving one bit for sign



So, let us start thinking in a way that how do we represent it in a decimal numbers. A negative number in a decimal representation, we represented by putting a negative sign. So, can we put the same negative sign in binary numbers? Possibly no, because whenever we are going to use binary numbers they would be in a digital world and in a digital world, digital world only understand the language of 0s and 1s. So, that means my negative number or that negative sign has to be represented in form of 0 or 1. So, let us think of some strategies how we can represent a negative number using 0s and 1. So one possibility, one naive possibility I would say would come to our mind that we can represent a, we can leave one particular bit as a sign bit.

(Refer Slide Time: 01:55)

Sign magnitude negative numbers



- MSB is designated to show sign. 1 represents negative, 0 represents positive.
- Example (8 bit binary number)
 - $(1001\ 0011)_2 = (-35)_{10}$
 - $(0111\ 1111)_2 = (127)_{10}$
- Range for 8 bit number – $(127)_{10}$ to $(-127)_{10}$
- Two zeros $(1000\ 0000)_2$ and $(0000\ 0000)_2$

Digital Logic Design-Introduction.

35




So, that kind of numbers I will call sign magnitude negative numbers. So, here 1 bit which bit to be choose, so let us say most significant bit. Let us say most significant bit is designated now to represent a sign bit and 1 would represent negative and 0 would represent positive. So, this could be one easy sign representation.

If we see this method, this is good. So, let us say we have a 8 bit binary number and I want to represent minus 35. So, if I want to represent minus 35, so I can put 8th bit or most significant bit or left most bit as 1 and remaining 7 bits I can use, remaining 7 bits I can use to represent 35, fair enough. So, 35 means 16, 32 plus 3 plus 1, 32 plus 2 plus 1 is 35, so I can represent minus 35 using this and similarly if I have to represent any number like positive number 127, so my most significant bit is 0 because the number is positive and the rest of the bits I can use to represent that number.

So, with this way we can clearly see the range of number could be minus, for 8 bit number it could be minus 127 to positive 127. So it looks good, intuitive and it looks to work also, but there are few challenges here. One challenge is that there are two 0s, one that in, this is a negative 0 because the sign is minus. 1 represent minus sign, so this particular representation is a minus 0 and this particular representation is a plus 0. So, having two 0s is complicated. We will see how. So, the other thing is, let us say I need to do some arithmetic here.


(Refer Slide Time: 04:08)



Sign magnitude addition/subtraction

- Addition
 - If signs are same → add
 - If signs are different → subtract
 - $(1000\ 1111) + (0111\ 0000)$
 - $= 111\ 0000 - 000\ 1111$ (subtract without sign bit)
 - $= 110\ 0001$ (result without sign bit)
 - $(1000\ 1111) + (0111\ 0000) = (0110\ 0001)$
 - Result's sign will change if number to be subtracted is bigger
- Subtraction
 - If signs are same → subtract
 - If signs are different → add

Digital Logic Design-Introduction. 36



So, what I need to do is, I need to see whether a sign bit is same or not. Let us say I want to do addition, if sign is same then I have to add if signs are different then I need to subtract. Similarly, if it is a subtraction then first I need to change the sign and then figure out whether it need to be added or subtracted. So and addition and subtraction otherwise can be done like we remove the sign bit, so how the process would be because sign bit have a special meaning here and the rest of the number is a magnitude number.


So, if I need to subtract these two numbers or add these two numbers, so this is a negative number and this is a positive number. So, what I need to do is, I need to subtract this positive number from the negative number. So essentially here also, I need to see that which number is a bigger number, if magnitude of this number is big, magnitude of this number is here, magnitude of this number is big, so that is why we are subtracting this from this, this number from this number.

And subtraction we have learned in our previous lectures, so we do the subtraction that way and then we get this result. So, whatever result we get after that we add the sign bit, so sign bit would be added based on the calculation, so because whatever number we found is a bigger number, so sign of that number will prevail. So, this is how subtraction and addition would happen if the signs are different.

So, here now finally we have added the sign bit of 0 because this number was a bigger number. So, there are couple of steps involved, it overall, in summary I can say this steps are first I need to find out which, I have to first remove the sign bit and then find out which number is a bigger number. If subtraction need to be done then we need to do subtraction using the subtraction method.


And then after that whatever result is achieved then we have to find which is the sign bit of the result and then catenate that or add that result bit in the MSB and then that is the final addition. Similarly, if subtraction has to be done then again if signs are different then we need to subtract, if signs are same then we need to subtract, if signs are different then we need to add and then find the corresponding result of the sign bit.

(Refer Slide Time: 07:17)




Sign magnitude summary

- Two zeros create confusion
- Addition may result in addition or subtraction
- Hardware implementation
 - Requires both adder and subtractor
 - Requires controller that determine which hardware to use
 - Separate handling of sign

Digital Logic Design: Introduction.37

So, overall process is little bit clumsy because here we are doing a, like we first of all require a different hardware for addition and subtractions and we do not know which hardware is required, you need to require both adder, subtractor and a controller which is determining when to do addition and determining which sign bit is required that is additional steps. So, along with this arithmetic confusion there is also additional confusion because of two 0 bits, so two representation of 0. Can we have something better? So, let us think of some other new representation.


(Refer Slide Time: 07:52)



Alternative – 1's compliment

- Assume
 - For negative number $-u$ flip all bits of $+u$
 - Nth bit represent sign (1 for -ve, 0 for +ve)
 - (N-1) bits represent magnitude
- Example (8 bit)
 - $(+13)_{10} = (0000\ 1101)_2$ $(-13)_{10} = (1111\ 0010)_2$
 - $(+69)_{10} = (0010\ 0101)_2$ $(-69)_{10} = (1101\ 1010)_2$
- Range -2^{N-1} to 2^{N-1}

Digital Logic Design: Introduction. 38




So, let us say we are representing it in a 1s complement manner. 1s complement means we are taking in complement of that number. Let us say if u is a number and I need to find out the negative of that u , then what I do is, I will flip all the bits. That means if the bit was 0, I make it 1, if bit is 1, I make it 0, so that way flip number is a negative number that is my assumption and this particular representation I am calling 1s complement representation.

So, because for positive number we assume that MSB most significant bit is 0, so because we are flipping, so last bit or most significant bit is 1, that means it is negative and if most significant bit is 0, then it is positive and N minus 1 bit would represent magnitude, so in a way this is very similar to sign magnitude, that 1 bit is reserved for sign and rest N minus 1 bit is representing magnitude but the order is different, because we are here inverting the bits.

So, let us take some example, let us say this is plus 13, 0000 1101 it is in 8 bit and then minus 13 it is simply inverting all the bits, 1111 0010 or similarly any other number, so essentially receiving or getting a negative number is easy I just need to flip all the bits or reverse the polarity of all the bits and because I can represent N minus 1 bits of magnitude, the range of the numbers would be minus 2 is to power N minus 1 to 2 is to power N minus 1. So, I can represent minus 1 to 2 is to power N minus, so for example if it is a 8 bit number I can represent minus 127 to 127, so this is the number of bits I can represent.

(Refer Slide Time: 10:01)




1's complement

- Mathematical background of 1's complement
 - $2^N - 1 = 111\dots1$
 - $-u = \sim|u| = 2^N - 1 - |u|$
- Issues
 - Two zeros (0000 0000) and (1111 1111)
 - Addition and subtraction is challenging

$$0000\ 1101 + 1111\ 0010 = 1111\ 1111$$
$$0000\ 1101 + 1111\ 0011 = 0000\ 0000$$

Digital Logic Design: Introduction. 39



Here also, we see that we have two 0s, so before that let us also see the mathematical background, so when I am flipping all the bits, it is also equivalent to subtraction from 2 is to power N minus 1 because 2 is to power N minus 1, so let us say the case is 8 bit number, for 8 bit number, the 2 is to power N will be 256, 256 minus 1 would be 255, so that means all 1s.

And if I want to get a negative number, I am essentially doing inverting of all the bits which also means I am subtracting from 2 is to power N minus 1 and I am subtracting this u or basically magnitude of this u from the 2 is to power N minus 1, so this can be we can say this is the mathematical representation of this number, negative number.

Now here also there are two issues, one issue is I have two 0s, one I call positive 0 because sign bit is positive and the other is the negative 0 which is invert of all this 0, so this is a negative 0. Addition and subtraction, if I am doing it on only positive numbers then it mostly works, if I am doing only with the negative numbers then also it works, but I am doing addition and subtraction using regular arithmetic or regular binary additions, then if one number is positive, one number is negative, it does not work.


So for example, here if you see this particular number is of 13 and this particular number is 1, if I invert this, this is 1101 so that means it is also 13. If I am subtracting 13 and adding positive 13 and minus 13, if I add plus 13 and minus 13, let us see the output 1 plus 0 is 1 0 plus 1 is 1 and

similarly all the 1s I am getting. So, minus 13 plus 13 is equal to minus 0, still okay. Still okay because at least it is giving me 0.

But on the other hand let us say, consider another number, so this number is minus, this number is plus 13 and this number is minus 12, so I am adding minus 12 and plus 13, the output is 0, so that means there is this particular output is wrong. So, addition works for a negative number or like adding two negative numbers or adding two positive numbers, adding negative and positive numbers will mostly work but it does not work in some cases.

So, because it does not work in some cases, this particular condition is not good at all. Because we believe computers, computer is supposed to do everything correct functionally. So, that means we need to find out some other representation which is better than this, which does not create this confusion and this particular error came essentially because we have two representation of 0s, one representing all 0s, another representing minus 0 that means all 1s. So, because of two representation of 0s we are getting this, this correction error of 1 in some cases. So, what can be the other method of representing negative numbers?

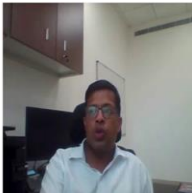
(Refer Slide Time: 14:10)



Another alternative – Bias

- Add a bias to number and find binary
- $F(u) = u + \text{bias}$
- Say bias for 8 bit number is 127
 - $(1)_{10} = (1000\ 0000)_2$
 - $(-127)_{10} = (0000\ 0000)_2$
 - $(+128)_{10} = (1111\ 1111)_2$
- Adding two numbers in binary
 - $F(u + v) = F(u) + F(v) - \text{bias}$

Digital Logic Design: Introduction.

40

Another alternative we can think or is we can add a bias. So everything is positive in our numbers, so let us say if I want to represent any number, I add that particular, some particular bias and for example, let us consider that for 8 bit number, I add a bias of 127. So, if I am adding a bias of 127, let us see how it works.

So I want to represent 1, I want to represent 1 in binary, so first thing I need to do is I need to add 127, so it become 128 now I will represent this 128 in regular binary, so that means 100 0000. So, similarly for any other number if I want to add, I want to represent in binary, then first I need to add this offset or this bias. For example, I want to represent minus 127, so first I need to add bias, so minus 127 plus 127 will be 0.


So, that means if this number is there 0000, it actually represents minus 127 in bias mode, so in a bias representation. Similarly, if I want to represent plus 128, the output would be 128 plus 127 that means 255 and 255 can be represented using 1111 1111. So, there are couple of observations here, now if I want to represent any number using bias method, then for every like if my number is 6 bit, 7 bit, 8 bit, bias has to be defined, it should be known to everybody who is doing computation.

So, it is better to standardize a bias if I am using a bias method of representing negative numbers. The other thing is, for representation sake it is still okay but if I need to add or if I need to subtract numbers, then I need to take care of bias. Because let us say I am adding two numbers, so $F + u + v$, u and v are two numbers, I want to add two numbers, so biased number this $F + u$ is bias number, this is also bias number because bias is added here as well as here. So, if I want to add two numbers, I need to finally subtract bias out of it.

So, this representation using biased method even for addition or subtraction, I have to, in case of subtraction I have to add bias, in case of addition I need to subtract bias, then only I can get actual addition or subtraction. The other thing is let us say the number is 6 bit, number is 5 bit, for 5 bit number bias would be different, for 7 bit number or 8 bit number bias is going to be different. So, this bias further, as I said earlier it should be known whenever we are doing any kind of arithmetic using this biased negative numbers.


This also looks crude, although we have solved one particular issue with this, that 0 will have only single representation, but because addition, even addition and subtraction would require additional step of adding bias or removing bias. For multiplication, it is even more difficult, we will not be able to use this bias method for doing multiplication. So this leave it, leave us at a crossroad that we are not able to find intuitively what, how should we represent negative numbers.

(Refer Slide Time: 18:38)



Summary so far

- Representation should be simple
- Two values of zero are not desirable
- Addition/subtraction should be easy



41

41

Digital Logic Design: Introduction.

So, to find out the solution let us go back and see that where why it is so challenging or what are we actually looking for? What should be the ideal representation of negative numbers? So, one this we would like is, it should be simple, two representation of 0s will give unnecessary hassle and the results are not desirable because it will create at least some confusion. The other thing the addition and subtraction should be easy and it would be because we are talking of a digital design or a digital hardware, it would be even better, if addition and subtraction both can be done using a same hardware.

Or in a same method it because we are sometime adding numbers, some of them are positive, some of them are negative, adding negative and positive numbers, so that is why if there is only single method of addition and subtraction that will make our life easy. So, again to look at things let us, let us look at another representation of numbers that may help us in understanding how should we represent negative numbers.

(Refer Slide Time: 20:03)

Number circle

<https://thesis.laszlokorte.de/jemo/number-circle.html>

Introduction.

42

So, this particular method of representing number is called number circle. So let us say we have 4 bit, I want to, I want to find out 4 bit numbers, so I would like to represent I can represent it like a circle. So, how do I write this, create this circles, because I want to create it using a 4 bit, so I will first of all write four 0s that will be the top of this circle and every increment will give me in a clockwise direction next number.

So, increment from 0 to 1 is next point in the circle, 1 to 2, 2 to 3, similarly, every addition is giving me a circle and when I reach 15 and again add one then it will become 0, so there is a discontinuity here. So, this discontinuity is also called breakpoint and if this essentially, this also means that I cannot represent any number bigger than 15 in a 4 bit representation. So, this is also called overflow.

So, there are couple of more things here, first of all I am moving in a clockwise direction. If I want to add, let us say if I want to add 4 to any particular number I want to add 4 to 1, so I will go four steps in a positive direction, 1 2 3 4 and let us say I want to subtract, let us say I want to subtract 4 from 12, what I will do is, I will go four steps in anticlockwise direction. So, in anticlockwise direction I go, so 11 10 9 8, so after four steps I got 8, I got at 8 so this is what it is.

But if I cross this particular line which is a line of discontinuity, then result will not be correct here. Here, at least in the positive circle, the results will be because we are reaching a condition of overflow.


So, let us say I want to add 4 to 14, so because I have crossed this particular circle 14 15, 0 and 1, so I will get an output of 1 but because it is crossing this particular line of discontinuity or line of breakpoint so I would get an overflow condition or similarly going in anti-flow, anticlockwise direction can give it, give me a underflow condition. So, this particular diagram has been taken from this website which we will visit after couple of slides. So, can I take some clue from this number circle.

(Refer Slide Time: 23:06)

Number circle

<https://thesis.laszlokorte.de/demo/number-circle.html>

Digital Logic Design Introduction. 43




If I represent in such a way that let us say, so that I can represent some negative numbers also. So in case of negative numbers, what I am doing is I am trying to do in the same way that going clockwise will give me correct results, going anticlockwise will also give me correct results.

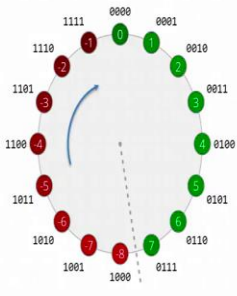
So, let us say I want to subtract 4 from 2, I want to subtract 4 from 2, so I will go in a negative direction or in a anticlockwise direction four steps, 1, 2, 3, 4, so subtracting 4 from 2 is giving me minus 2. Similarly, if I am adding 5 from minus 3, I am adding 5 to minus 3 then I am going in clockwise direction because I am doing addition, I am going five steps, 1, 2, 3, 4, 5, so adding 5 to minus 3 is giving me 2. So this means, this number circle, if I represent in this way it can give me some good characteristics that it is a intuitive or a standard representation.

(Refer Slide Time: 24:23)

Number circle



Clockwise – addition
Anticlockwise – subtraction
Positive number
- u clockwise steps from 0
Negative number
- u steps anticlockwise from 0
u clockwise steps =
 $2^N - u$ anticlockwise u steps



<https://thesis.laszlokorte.de/demo/number-circle.html>

44

So, we understand clockwise means addition and anticlockwise is subtraction. So, this is at least one observation we got from here, the other observation we can see is if I want to see any, I want to get a representation of plus a number or minus number, so let us say I want to see plus 4, so what I can do I can do write clockwise direction from the 0 1 2 3 4, it gives me this particular number.

So, that means this is the representation of plus 4, so similarly if I go in a anticlockwise direction from the 0 then it will give me minus 4, minus 1, minus 2, minus 3, minus 4, so if I put my numbers in this way, so we are all talking, I am right now talking only in the decimal form, in the decimal form along with the decimal form, we can look at the binary also, so that means 1111, adding 1 will give me 0000 and this addition is all correct.

But adding 1 from 1101 will give me 1100 and if I add 1 to this number then it would be 1101, so that means again it is a natural representation that adding 1 in a clockwise direction is giving me correct results and there is only one representation of 0, discontinuity would happen here, so discontinuity would happen here, so if I am crossing this particular line, so let us say I am adding 1 to 7, then I will give me 1000, because I am crossing this line discontinuity then this would be an overflow condition.

So, because 1000 does not represent 8 but it represent minus 8. 8 is essentially out of bound for this particular representation, for 4 bit number, the number range is a 7 or minus 8 to 7. So,

whenever I cross this particular line of discontinuity or a line of breakpoint, then there would be overflow or underflow; overflow if I am going in a positive direction and underflow if I am going in a anticlockwise direction.

The other observation here is, let us say I am taking u steps or some x steps in clockwise, let us say I am taking four steps from 0 to four steps here, this also means the same thing can be achieved by going anticlockwise in twelve steps or $2^N - u$ anticlockwise steps are equivalent to u clockwise steps. So, let us remember again, clockwise means we are adding u numbers, adding a number u and anticlockwise means we are subtracting number u .

So, that means if I can represent addition and I can represent it using subtraction also that means here addition can be represented using subtraction that is a very magical property, this particular circle has. So, because now addition can be represented using subtraction, so that means I can do addition using subtraction or subtraction using addition, this makes my life simple, easier, there would be only single representation and there would be only single way of adding numbers or subtracting numbers. So, I will quickly break this presentation to see this number circle and then we will get back to this presentation.