

Digital IC Design
Prof. Janakiraman Viraraghavan
Department of Electrical Engineering
Indian Institute of Technology, Madras

Lecture – 60
Two's Complement Arithmetic

So, today we will proceed into the discussion on multipliers ok. So, in this class I will just cover the basics of what you know the multipliers are, you know what this terminologies are and I will also go into the 2s complement number representation right. So, how many of you are already aware of 2s complement arithmetic, not 2s complement representation I mean 2s complement arithmetic right, when you do a when you do a multiplication what should you do and all that.

So, have you worked out the math of why for example, sign extension will retain the value? Yeah, you guys? You have not worked out that math? So, that we will do in this class so that you get a hang of you know the mathematics also behind it. So, that we are perfectly consistent with what we are doing in the rest of the module ok. So, fine.

(Refer Slide Time: 01:21)

28/10/2019

EES311

MODULE-6 - MULTIPLIERS

$X \rightarrow M \text{ BIT NUMBER}$
 $Y \rightarrow N \text{ BIT NUMBER}$

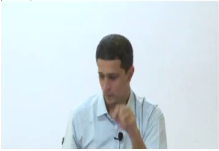
} UNSIGNED

$\Rightarrow X = \sum_{i=0}^{M-1} x_i 2^i$
 $Y = \sum_{j=0}^{N-1} y_j 2^j$

$Z = XY$

$\sum_{k=0}^{(M+N-1)} z_k 2^k = \sum_{i=0}^{M-1} x_i 2^i \times \sum_{j=0}^{N-1} y_j 2^j = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} (x_i y_j) 2^{i+j}$

$z_k = \sum_{(i+j=k)} x_i y_j$



So, I have X which is let us say an M bit number and Y which is an N bit number and I am going to assume that both these numbers are unsigned. What does this mean? It means that X can be written as $x_i 2^i$ where i goes from 0 to M minus 1. Similarly, Y can be written as $y_j 2^j$ where j goes from 0 to N minus 1.

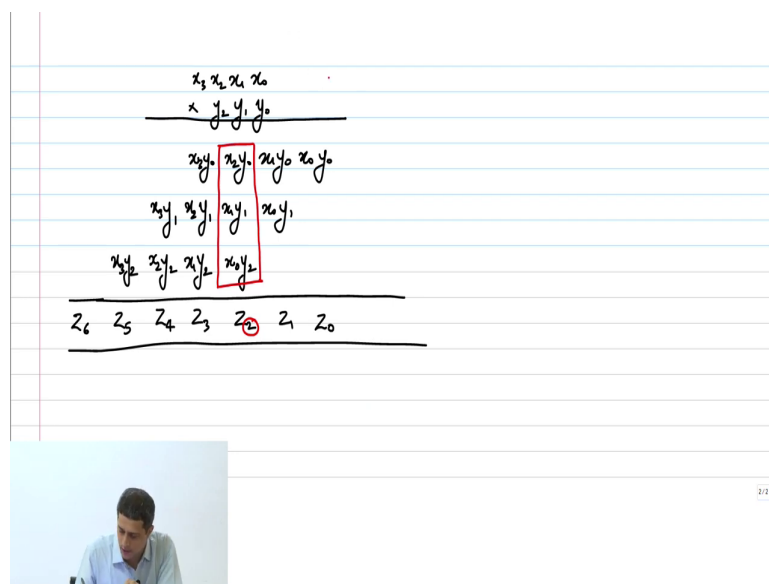
So, I am now going to create a product of these two numbers and this will be X times Y. So, if X is M bits and Y is N bits how many bits will Z be? So, we will look at that. So, this is equal to $\sum_{i=0}^{M-1} x_i 2^i \times \sum_{j=0}^{N-1} y_j 2^j$. This can simply be written as $\sum_{i=0}^{M-1} \sum_{j=0}^{N-1} x_i y_j 2^{i+j}$ and I am now going to write this as a product $\sum_{k=0}^{M+N-1} z_k 2^k$.

K is going to go from 0. So, if you take the 0 i equal to 0, j equal to 0 right, clearly i plus j will also be 0 there. So, it starts from 0 and what will be the maximum value? M minus?

Student: (Refer Time: 03:40).

M minus M plus N minus 2, but you will find that there is also a carry that comes because you are doing an addition and you could get at most one extra bit there. So, this will become M plus N minus 1 $2^{\text{power } K}$ ok. So, what is this coefficient $2^{\text{power } K}$? This is equal to summation of $x_i y_j$ such that i plus j equals K. So, you are just grouping all the terms and you are able to evaluate the coefficient of $2^{\text{power } K}$ ok.

(Refer Slide Time: 04:35)



$$\begin{array}{r}
 x_3 x_2 x_1 x_0 \\
 \times y_3 y_2 y_1 y_0 \\
 \hline
 x_3 y_3 \quad x_3 y_2 \quad x_3 y_1 \quad x_3 y_0 \\
 x_2 y_3 \quad x_2 y_2 \quad x_2 y_1 \quad x_2 y_0 \\
 x_1 y_3 \quad x_1 y_2 \quad x_1 y_1 \quad x_1 y_0 \\
 x_0 y_3 \quad x_0 y_2 \quad x_0 y_1 \quad x_0 y_0 \\
 \hline
 z_6 \quad z_5 \quad z_4 \quad z_3 \quad z_2 \quad z_1 \quad z_0
 \end{array}$$

Handwritten polynomial multiplication showing the result of multiplying two polynomials. The result is shown with coefficients $z_6, z_5, z_4, z_3, z_2, z_1, z_0$. A red box highlights the terms contributing to z_2 : $x_2 y_0, x_1 y_1, x_0 y_2$.

So, let us look at a simple example $a^3 a^2 a^1 a^0$ times $b^2 b^1 b^0$ ok. So, my x is or let me give the notation consistent $x^3 x^2 x^1 x^0$ $y^2 y^1 y^0$ ok.

So, what do you get here the first term will be $x^3 y^0$ $x^2 y^1$ $x^1 y^2$ and $x^0 y^3$. What is the second term? It is basically just a shifted thing right you have to shift this term by 1 bit and add it is a shifting and adding process that we have to do here. So, this is and this is exactly how we do normal multiplication right. So, this will be $x^1 y^0$. Sorry $x^1 y^0$, no I need to do x^0 into y^1 right. So, that is $x^0 y^1$ right and then $x^1 y^1$ $x^2 y^2$ $x^3 y^3$. The last term is $x^3 y^0$, I am sorry. $x^0 y^2$ right and $x^1 y^2$ $x^2 y^2$ $x^3 y^3$ y^2 sorry, ok. So, what you notice here is this is going to be my Z^0 Z^1 Z^2 Z^3 Z^4 Z^5 and in doing the addition of $x^3 y^1$ $x^2 y^2$ you could get a carry and that could generate a carry and you will get Z^6 , clear?

Now, what do you notice here? This 2 is nothing, but addition of all the terms whose product is such that i plus j is 2. If you look at the $x^2 y^0$ $x^1 y^1$ $x^0 y^2$ it is basically the sum is always 2 right that is how we that is how we grouped it in the earlier summation as well, clear? So, what we notice here is let me get this terminology right I always get confused with this.

(Refer Slide Time: 07:57)

Multipliers - Definitions

			1	0	1	1	Multiplicand	
			×	1	1	0	Multiplier	
				0	0	0	0	PP0
			1	0	1	1		PP1
		1	0	1	1			PP2
1	0	0	0	0	1	0		Result

Multipliers need to perform three main tasks:

- ▶ Partial Product (PP) Generation
- ▶ Partial Product Accumulation
- ▶ Final addition

Address: 171M EECS111, Parallel IP Design: Module 6 - Addition and Multiplication 10/10

This is called the multiplicand, this is the multiplicand and this is the multiplier and in the process you generate what is known as partial products. These are essentially partial products ok. So, this is PP0, PP1 and PP 2. So, if I if x is M bits and N and y is N bits how many partial products will I have? Yeah? N, right? basically as many as there are bits in y, as many as there are bits in the multiplier right. So, this is basically you will have N partial products let me call it PP right. And, what is the size of each partial product? N bits M bits sorry and each will be M bits long.

So, in order to perform multiplication you need to do the following tasks. One is partial product generation ok, then you need to do partial product accumulation or let me put one more term there partial product shifting, you have got to shift an add right. This is one more operation that needs to be done. What is this? Third operation is partial product accumulation.

In order to speed up the multiplier, you need to speed up all of these operations in order to get better performance ok.

So, there is also one more operation that needs to be performed that is called the final vector addition. You will see why that needs to be done when we look at a actual implementation using full adders ok. So, for partial product generation what do you think is the logic operation that needs to be performed and right. So, this is nothing, but a AND gate. What about partial product shifting yeah? So, this is a slightly tricky thing and not tricky actually, it is actually very trivial once you implement it and see. As you say yeah if you want to do left shift you have to move everything by this thing, but that can be simply achieved by wiring ok.

So, let us come to the next thing what do you need in order to do partial product accumulation? You need an array of full adders. We have to arrange this so that we can perform this operation here right. This you clearly see there is an array of terms that need to be added in a particular way and you need to provide as many full adders as needed to perform these many additions there right. So, this is nothing, but an array of full adder.

Now, once you provide an array it turns out that the previous operation of partial product shifting is very trivially achieved by simply wiring ok. You do not you do not have to do in the sense all you have to do is ensure $x_1 y_n$ gets added to $x_n y_1$. So, if you just wire it correctly you will find that the shifting operation is already done very easily ok. This final vector addition we will come to later because we have it is not evident from this figure here ok. So, we will look at it later and this is where your architecture of these other adders come into picture.

This previous one partial product accumulation can be done using a typical ripple adder just a full adder you ripple it through do it nicely you will get it ok, but this final vector addition if you put a ripple adder it will kill your delay. So, here you have to use some fast adder architecture and this is where I am suggesting that you use something like a carry look ahead adder to speed up your multiplication ok.

Now, what we discussed was only for unsigned numbers. Yeah? Yeah, we will come to it when I show you the implementation using full adders; you will find that there is one final addition that needs to be done ok. In the sense you put all these you put these arrays together put enough number of full adders suppose you are doing 4 cross 4 multiplication you put 16 full adders in this so that you can do this addition appropriately multiplication addition appropriately.

You will find that the last stage just turns out to be a ripple and if you just put full adders that last stage is just like a ripple adder. You can speed up that process by making that a faster more intelligent adder is all I am saying. One of the last rho in the array happens to be your ripple adder, other things are not ripple adders they are full adders, but they not a ripple adders we will see why. Yeah, actually in reality if you look at a microprocessor nobody cares if you are doing 4 bit into 3 bit everything will be converted into 4 bit into 4 bit you happen zeros and proceed right that is all.

So, this N and M is only to tell you know just tell you which one is giving rise to more partial products which one is giving rise to you know that kind of thing.

No, that is why I said if we if you are doing x into y like this then number of partial products will be this right if you do it the other way then here you will have more partial products ok. This is the convention I am saying M is x, x is M bits, y is N bits ok.

(Refer Slide Time: 15:51)

SIGNED NUMBERS


2's COMPLEMENT REP

$$\{a_{N-1} a_{N-2} \dots a_0\} \rightarrow \sum_{i=0}^{N-1} a_i 2^i \text{ --- UNSIGNED}$$

$$\downarrow$$

$$-2^{N-1} a_{N-1} + \underbrace{\sum_{i=0}^{N-2} a_i 2^i}_{(2^{N-1} - 1)}$$

$$\sum_{i=0}^{N-2} a_i 2^i \leq \sum_{i=0}^{N-2} 2^i = (2^{N-1} - 1)$$

$$-2^{N-1} + (2^{N-1} - 1) = -1$$


So, unfortunately what we did earlier was just unsigned numbers that is very clear. So, we have to also deal with sign numbers ok. So, I want to represent negative numbers and therefore, the most obvious thing to do is to assign that most significant bit to a sign bit. If it is 1, the what the most trivial way of doing it is if that bit is 1 is treated as a negative number the remaining N minus 1 bits as a negative number or treat it as a positive number. Problem with this is the 0 happens to have two assignments plus 0 minus 0 which is useless.

And, therefore, people move to this thing of 2s complement representation. 2s complement representation is I have an N bit number a_{N-1} , a_{N-2} , all the way to a naught right. If it were an unsigned number this would simply be value would simply be i equal to 1 to N minus 1 $a_i 2^i$ if it were unsigned. correct? But, if it is a sign number now 2s

complement representation is the following I will write this as $-2^{N-1} + \sum_{i=0}^{N-2} a_i 2^i$.

So, how do you identify now if this number is positive or negative? If I give you a 2s complement number you look at the MSB. So, if the MSB is 1 you are saying that the number has to be negative, how do you know?

Student: (Refer Time: 18:10).

Yeah, in some sense the maximum value of this guy is $2^{N-1} - 1$. This term is less than or equal to this. So, what is the maximum value of that remaining part $\sum_{i=0}^{N-2} a_i 2^i$. I'm sorry this is $0 \leq \sum_{i=0}^{N-2} a_i 2^i < 2^{N-1}$. So, let us look at that $\sum_{i=0}^{N-2} a_i 2^i$. What is the maximum value what configuration of bits of a $\sum_{i=0}^{N-2} a_i 2^i$ will give you the maximum value? all 1 because is all positive numbers right.

This is less than or equal to $\sum_{i=0}^{N-2} 2^i$. Therefore, this is less than which is actually equal to $2^{N-1} - 1$ right. Now, clearly if the MSB is 1 that is a $-2^{N-1} + \sum_{i=0}^{N-2} a_i 2^i$ then the number right is going to be $-2^{N-1} + \sum_{i=0}^{N-2} a_i 2^i$. Therefore, if that MSB happens to be 1, then number has to be a negative number, it cannot be a positive number ok.

So, now, if it is a negative number what do you do in order to find out its value? Yeah? No 2s complement representation is 1, what is the other thing that you do? Your taught and algorithm right of generating the value take complement of every bit. Take 1s complement, add 1 to it we will see why that has to be true ok.

(Refer Slide Time: 20:23)

if $a_{N-1} = 1$; THEN THE NUMBER = $\left(\text{1's COMPLEMENT} + 1 \right) - \text{ve.}$

$1101 =$
 $-2^3 + 2^2 + 0 + 2^0 = -3$ $\{ a_{N-1} a_{N-2} \dots a_0 \} + \{ \bar{a}_{N-1} \bar{a}_{N-2} \dots \bar{a}_0 \}$
 $= (1111 \dots 1) + 1$ (N times)

1101
 $\rightarrow 0010 \rightarrow 1111 - \{ a_{N-1} a_{N-2} \dots a_0 \} = (11 \dots 1) - \{ \bar{a}_{N-1} \bar{a}_{N-2} \dots \bar{a}_0 \}$
 $+ \quad 1$
 $= -1 - \{ \bar{a}_{N-1} \bar{a}_{N-2} \dots \bar{a}_0 \}$
 $= -(1 + \{ \bar{a}_{N-1} \bar{a}_{N-2} \dots \bar{a}_0 \})$
 $= -\text{ve}(0011)$

So, now if a N minus 1 equal to 1 right then the number is equal to 1s complement plus 1 and then you put a negative value of it is that is what they tell you right. So, for example, if I have a number 1101 and I tell you that this is a 2s complement number. First of all what is the value of this number? Yeah? minus 3 yeah why is that because this is a 4 bit number. So, this is minus 2 power 3 right I can let me write it here minus 2 power 3 plus 2 power 2 plus 0 plus 2 power 0 right. So, which is equal to minus 3.

Now, what are we saying if I want to this is from the 2s complement evaluation, if I do the algorithm that I just told you 1s complement 1101 if I complement I will get 0010 and then I add a 1 I will get 0011 right. So, the answer is minus of 3, negative of this number ok. So, let us see formally why this thing has to be true. So, if I take my N bit number which is in 2s complement form a N minus 1 a N minus 2 all the way to a naught right and I just add it with

a $N-1$ bar a $N-2$ bar all the way to a naught bar. What does this be equal to 1111 every bit I am adding with it is complement correct.

So, this can be no carry first of all because this is one that will be 0 if this is 0 that will be 1. So, this will be 1111 all the way and how many 1s? N times right now what are we saying a $N-1$ a $N-2$ this representation value is equal to 1111 N times, what is that value? 2^{N-1} correct no, wait we will we will come to read we leave this as it is for now minus bar a $N-2$ bar N all bar, correct?

So, now what have we done here 111 minus 1 first what do we ok. So, that this number is a $N-1$ bar all the way to this right sorry no wait. So, what do we want to do we want to basically take the number complement every bit and then show that we have to add a 1 right? So, this is nothing, but 2^{N-1} this is going N times right.

What is this one here this is nothing, but actually the 111 minus the number itself a 3 a 2 a 1 a naught correct? That is what it is; then we are doing a plus 1 right. So, what do you get why should you do that plus 1 is the question? Yeah.

You are saying that that value is nothing but minus 1 in 2s complement representation?

Student: 1's complement (Refer Time: 26:13).

1s complement of that yes that is right. So, this is minus 1 minus a $N-1$ complement a $N-2$ complement a naught complement. So, this is nothing but minus of 1 plus every bit complemented. So, therefore, this is a very consistent operation that you have been asked to do and it will definitely work.