Digital IC Design

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

Lecture - 62
Array Multiplier

So good evening. As I mentioned last class I go a little over time today; I want to complete up to the sign multiplier ok. So, I will have to cover 3 topics I think normally I have to cover 2; I just take a little bit of extra time maybe 20 minutes extra time to go and finish the third topic ok. And after today's class you should be all set to finish the next assignment also right. You know you should have all the information you can go ahead and implement do DRC by LVS lay out everything ok.

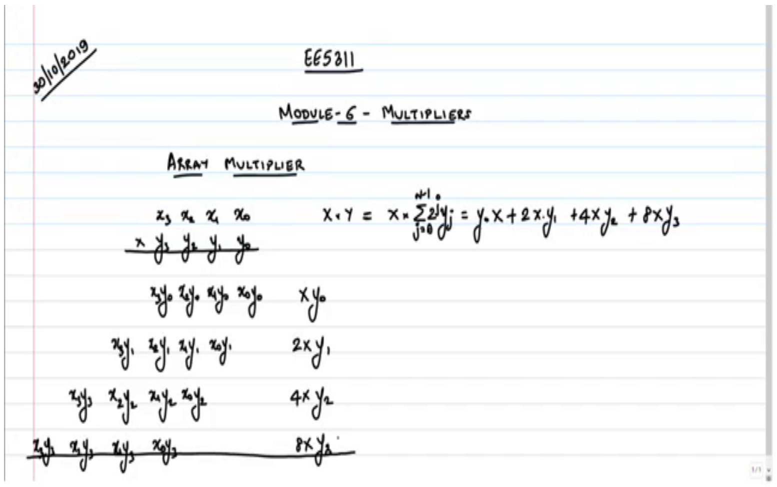
So, that is the aim of today's class so, please pardon me I would go over time a little bit. Hopefully I will not have to take an extra class somewhere; that is the aim. And before I start our friend has lost an iPad in this class last time Hemanth. So, if someone is playing a prank please return it; he is desperately looking for it; we are not able to locate it ok. So, if you find it or if you know saw someone playing a prank with it or something then please come up and just tell him right now ok; because he has a lot of information in it and of course, its also very expensive right.

So with that let us get started. So, in the last class we covered all the theory behind signed arithmetic right, how do how to deal with 2's complement numbers, how to deal with you know sign extension of 2's complement number. Then we look at some examples of how you can do multiplication of 2 sign numbers right. And there if you do not do sign extension properly you will not get the answer correctly. Then also the final carry out that you get from the circuit right, from this multiplication should be used to propagate as many sign extension bits as you want ok.

That is the use of we carry out in a sign multiplier in an unsigned multiplier the carry out forms a more significant bit. In a sign multiplier that carry out is not the most significant bit, it is only to help you do sign extension to as many bits as you want it and keep the answer consistently; this is a very key difference between the 2 multipliers.

So, very obviously, the way you do sign multiplication and the way you do unsigned multiplication are very different. Which means that, if I give you an n bit number unless I tell you whether this is a sign representation or an unsigned representation you cannot perform the multiplication and that is why in your microprocessor you have two different instructions called MUL and IMUL right. If you take the x 86 architecture, there is MUL and IMUL 1 is signed multiplication other is unsigned multiplication ok. And today we will see why that hardware also has to be different and how it has to be different ok.

(Refer Slide Time: 03:30)



30/10/2019

EE5311

MODULE-6 - MULTIPLIERS

ARRAY MULTIPLIER

$$X \times Y = X \times \sum_{j=0}^{n-1} 2^j y_j = y_0 X + 2X \cdot y_1 + 4X y_2 + 8X y_3$$

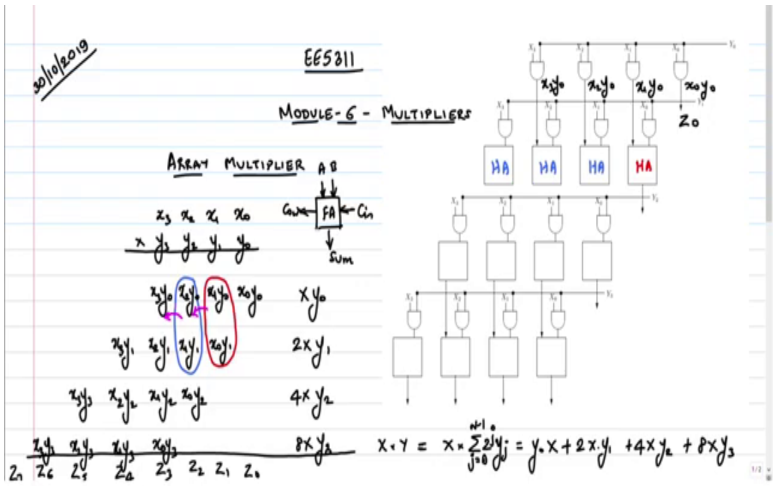
$$\begin{array}{ccccccccc}
 & & & & x_3 & x_2 & x_1 & x_0 & \\
 & & & \times & y_3 & y_2 & y_1 & y_0 & \\
\hline
 & & & & x_3y_0 & x_2y_0 & x_1y_0 & x_0y_0 & Xy_0 \\
 & & & x_3y_1 & x_2y_1 & x_1y_1 & x_0y_1 & & 2Xy_1 \\
 & & x_3y_2 & x_2y_2 & x_1y_2 & x_0y_2 & & & 4Xy_2 \\
 & x_3y_3 & x_2y_3 & x_1y_3 & x_0y_3 & & & & 8Xy_3 \\
\hline
\end{array}$$

So, let us start of an Array Multiplier. As the name suggests what am I going to create an array of? Full adders basically right. So, let us take a simple example x 3, x 2, x 1, x naught; y

3, y 2, y 1, y naught right. So, if you look at the partial products it will be like this; x 1 y naught, x 2 y naught, x 3 y naught then I want to shift and add ok. So, what is this? It is x naught y 1, x 1, y 1, x 2, y 1, x 3, y 1 ok. Then again another shift it is x naught y 2, x 1 y 2, x 2 y 2 and x 3 y 3 finally, you do one more shift and add. and what will you get? You will get x naught y 3 sorry, x 1 y 3, x 2 y 3 and x 3 y 3.

What is this by the way? This is basically X into Y right and if we are going to write Y as we know the in the binary representation it is 2 power j into y j right; j equal to 0 to n minus 1 right. So, if I write this as it is if I write expand it will basically be y naught into X plus 2 X into y 1 plus 4 X into y 2 plus 8 X into y 3; that is what this shifting operation is. This is basically X y naught X times y naught; this shift and add is basically 2 X times y 1 this is 4 X times y 2 and 8 X times y 3 ok. This is the shifting and adding part that we are doing here. So, now, my task is to basically place full adders so that I can accomplish this task yeah.

(Refer Slide Time: 06:29)


30/10/2019
EE5311
MODULE-6 - MULTIPLIERS
ARRAY MULTIPLIER
A B
Cout
FA
Cin
Sum
x_3 x_2 x_1 x_0
$\times$ y_3 y_2 y_1 y_0
x_3y_0 x_2y_0 x_1y_0 x_0y_0
x_3y_1 x_2y_1 x_1y_1 x_0y_1
x_3y_2 x_2y_2 x_1y_2 x_0y_2
x_3y_3 x_2y_3 x_1y_3 x_0y_3
z_7 z_6 z_5 z_4 z_3 z_2 z_1 z_0
Xy_0
$2Xy_1$
$4Xy_2$
$8Xy_3$
$X \cdot Y = X \cdot \sum_{j=0}^{n-1} 2^j y_j = y_0 X + 2Xy_1 + 4Xy_2 + 8Xy_3$
HA
HA
HA
HA
Z_0

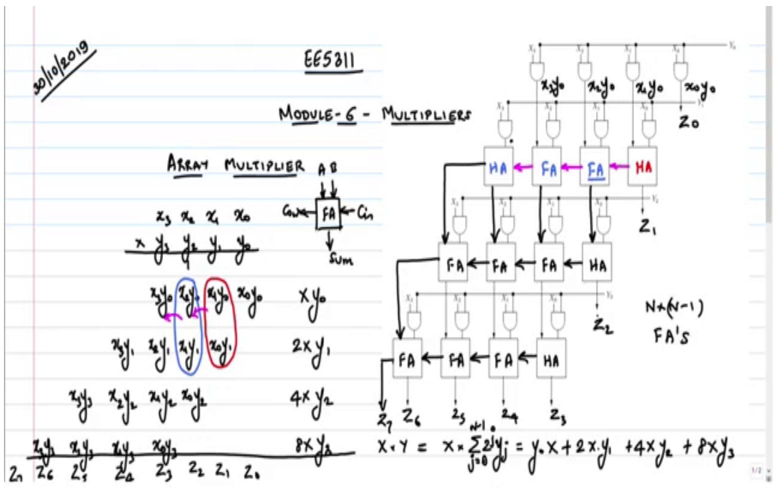
So, that is I can keep the multiplier in there. So, now, you should tell me what these blocks are ok, whether these are half adders or full adders. And how do I cascade them how do I connect them? So, this is z naught, z 1, z 2, z 3, z 4, z 5, z 6 and I have a z 7 also. So, you have to tell me where these z naught to z 7 variables are going to come out of the circuit and what these individual blocks are ok. Clearly, these are AND gates and they are creating my partial products x naught for example, this is what its x naught y naught yeah and this is x 1 y naught x 2 y naught x 3 y naught and so on right.

So, first of all where is z naught? This is z naught ok. Now, I need to perform which one? This addition x 1 y naught plus x naught y 1. So, what block should that be? Half adder basically because there is no carry right half adder. What about the other adders on that same row? They are all so for example, you have to perform x 2 y naught right and x 1, y 1 right this block if you look at it this is what we are performing here, so that will also be a half adder right.

So, everything here will be a half adder ok. So, you add these two numbers x 1 y naught x naught y 1 you get a carry right; that carry is now going to propagate from this red to this blue block correct. Because I need to add it to my next set of terms correct. So, what will be the carry out direction? Assume, that my half adder block is like this ok, it takes A B C in gives C out and sum sorry full adder half adder will be just grounded that is all there is no C.

So, what is the direction of flow for the carry out on the first row? Right. So, what we are doing is we are taking this carry propagating then we add those three numbers we will propagate like this correct. Or not even those three numbers I am adding those two numbers and then propagating correct. So, first of all can that be a full half adder at all in the way I am showing you here. Why?

(Refer Slide Time: 10:24)



Because I am propagating the carry therefore, this has to be a these blocks have to be full adder right and I am going to propagate my carry like this ok. Now where is, what is the output of this half adder in red? That is nothing but z 1 right, this is going to be my z 1; which one that is the final one can be because there is no other term coming in that is also true very good ok.

So, this can be half adder correct know, is right because there is no other term coming there ok. So, now, what about row 2? So, the first full adder in blue right this guy what did we do? We basically added x to y naught with x 1 y 1 and propagated the carry. What about that sum? That should get added to x naught y 2 correct.

So, therefore, the sum has to propagate like this correct. Now what is the is there a third input to this block? Yeah, that is good point, true no its not actually more adders I will come to that,

that is no the aim is not to reduce the number of adders. The aim is to do it as fast as possible. So, we come to that it turns out that by doing what you said you will actually speed it up by putting in more adders I will come to that, that is the next architecture.

Why not? Because I get x 1 y naught x naught y 1 I get a carry right, I am going to add that carry here right. So, basically I am adding these three numbers, no again the carry right from each addition that I get; I get a carry out I have to take it to the next stage. No, the sum the carry always goes to the higher bit, it goes to the 2 power 1 into something plus the sum, the carry does not affect the sum that goes to the next bit correct.

That will not affect the value of z 2 is what I am saying; because the carry is going to the higher bit; see this is 2 power 0 into z naught is the coefficient of 2 power 0 z 2 is the coefficient of 2 power 2 right, that carry that comes out of the addition of x to y naught x 1 y 1 on the carry is actually going to 2 power 3. So, therefore, it will not affect this z 2 which is the coefficient of 2 power 2 ok.

So, maybe see the carry might be 0; which is fine, but if the carry is 1 then it has to go to the next 1 right. Functionally this is fine, it turns out it is a different thing that timing wise it may not be very efficient ok. So, let us proceed, we will come back to your doubt, if it still exist after some time ok. So, now, the second block right again here the sum has to basically come down correct. What about the second block? What should that be the first one? Half adder again right, this is a half adder. This one? Full adder, full adder, half adder right you have to be careful here for now we keep it as half adder.

The last stage again half adder, full adder; sorry, full adder and half adder by the same logic correct. So, now what happens to the carry propagation again here, it is going to go like this right and of course, the sum has just come down like that ok. So, what is this term here? z 2 this is z 3, z 4, z 5, z 6 right. Now what happened to the carry out from those left corner half adders? I am adding x 3 y 1 here x 3 y 1 with the carry from the x 3 y naught x 2 x 2 y 1 term then I carry out come in, I add those 2 terms. Why does that carry go? So, that has to come here.

So, now, can this be a half adder? So, it has to become a full adder. What is the term you are missing now? z 7, z 7 that is coming out for me last full adder there right. So, this basically tells you how to line up array of full adders right; even the half adder I can consider the full adder rate. When you are trying to make an array structure you do not optimize on that area there, you take a full adder and repeat it.

I am just calling it half adder full adder; so, that it makes it more clear ok, otherwise they are basically just full adders. So, in order to perform 4 cross 4 multiplication how many half adders or full adders do I need? Yeah, n into n minus 1 right so, therefore, you need n into n minus 1 full adders effectively right.