

Architectural Design of Digital Integrated Circuits
Prof. Indranil Hatai
School of VLSI Technology
Indian Institute of Engineering Science and Technology, Shibpur, Howrah

Lecture – 36

Multiplier Architecture (Contd.)

So, welcome back to the course on Architectural Design of ICs. So, in the last class we have seen the architecture about Multiplier Design. So, how efficiently we can design multiplier so, that we have seen. So, we have seen signed multiplication and then unsigned multiplication, what are the architectures, some of the architectures are; that means, they have been used for area optimization and some of the architecture has been used for speed optimization.

So, that means, in some of the actually it depends upon the application; that means, for which application I am designing or I am intending to design the circuit or that particular multiplier. So, depending on that we choose different kind of architecture. So, and then we have seen that array multiplier architecture too where there is a regular structure; that means, the data flow on that particular structure it is very much regular. So, that type of architecture different kind of multiplier architectures we have already seen.

So, then another point is that suppose we need to do squaring. So, squaring is another that means that is another form of multiplication where we multiply with the same numbers. So, we have seen till now whatever is the number we have seen that means, the multiplier and the multiplicand they are 2 different numbers ok. So, considering that fact then we have gone for the different architecture for that.

Suppose the two; that means, multiplier and multiplicand both are same. So, at that time can I do more optimization on the circuit, so, that that it became hardware efficient. Hardware efficient in terms of that means, that area as well as the power as well as the speed, can I improve it if I am doing; that means, if I am multiplying the same number with it ok. So, that squaring circuit now we will investigate on this today's lecture.

(Refer Slide Time: 02:29)

Optimizations for Squaring (1)

Multiply x by x


	x_4	x_3	x_2	x_1	x_0
x	x_4	x_3	x_2	x_1	x_0
		x_4x_0	x_3x_0	x_2x_0	x_1x_0
	x_4x_1	x_3x_1	x_2x_1	x_1x_1	x_0x_1
	x_4x_2	x_3x_2	x_2x_2	x_1x_2	x_0x_2
x_4x_3	x_3x_3	x_2x_3	x_1x_3	x_0x_3	
x_4x_4	x_3x_4	x_2x_4	x_1x_4	x_0x_4	

x_0x_0 Reduce to x_0
 Move to next column

Reduce the bit matrix

	x_4	x_3	x_2	x_1	x_0
x	x_4	x_3	x_2	x_1	x_0
	x_4x_3	x_4x_2	x_4x_1	x_4x_0	x_3x_0
	x_4	x_3x_2	x_3x_1	x_3x_0	x_2x_0
		x_3	x_2	x_1x_0	x_1
			x_2	x_1	x_0

P_9 P_8 P_7 P_6 P_5 P_4 P_3 P_2 P_1 P_0



So, this is this squaring. So, suppose I have to multiply x with x . So, now, consider x are of 5 bits ok. So, x_0 to x_4 then the same number has been multiplied which is x_0 to x_4 . So, the just like the same multiplication algorithm whatever we have seen it is just bitwise multiplication initially. So, bitwise multiplication means, x_0 will be multiplied with x_0 , then x_0 into x_1 , x_0 into x_2 , x_0 into x_3 and x_0 into x_4 . Then again it will be whenever I have 2 that means I will go for this next bit, at that time it will be started from 1 bit left shift as the position is 1 bit shifted.

So, depending on the weight of this particular position now I have to start from here. So, just like the same I will get or I will get; that means, I will form the partial products. So, whenever we are now actually in multiplier what we do? Whenever, we got this partial product, so, after that what we do? That means, this column wise we just add it.

So, the different the addition method how I will add that is different depending on different architecture. But here, so, before to; that means do this addition of this partial products what I can do? That means, here if you just investigate or if you just closely look that x_0 into x_0 if I multiply x_0 with x_0 that will be nothing but your x_0 .

Why? Because if I multiply a with a according to the Boolean function logic, so that will be a . So, in the next column if you see what I am doing here? x_1 into x_0 , x_0 x_1 . So, this is nothing but x_1x_0 x_1x_0 . So, if I add x_1x_0 x_1x_0 twice, so, at that time what it will be? It will be 2 of x_1 and x_0 . So, 2 of x_1 and x_0 means, this position now this x

1 and x 0, this I can write at the next corresponding left hand side. Why? Because this is the position with weight of multiplied by 2.



So that means, if this is the weight binary weight for this particular position is 2, so, then this particular weight of this particular position that is 4. So that means, this 2 is multiplied with another 2. So, 2 of x 1 and x 0 means this x 1 and x 0 can be written here instead of this particular position ok. So that means, I do not need to add this x 1 and x 0; that means, I do not need any addition operation to or this ORing operation to calculate this x 1 and x 0 plus x 1 and x 0.

Just like the same I can remove or I do not require the AND gate which is basically doing this x 0 multiplied with x 0 why because x 0 into x 0 that is nothing but my x 0. So, the directly x 0 can be mentioned as P 0 or which is the; that means, the product 0-th product bit. So, in this way if I investigate this whole term whole partial products, I can get some of the optimization factors ok.

(Refer Slide Time: 06:24)

Optimizations for Squaring (2)

$\begin{array}{r} x_i x_j \\ x_j x_i \\ \hline x_i x_j \end{array}$	$x_i x_j + x_i x_j = 2 x_i x_j$
$\begin{array}{r} x_i x_j \\ x_i \\ \hline x_i x_j \end{array}$	$x_i x_i = x_i$
$\begin{array}{r} x_i x_j \\ x_i \overline{x_j} \\ \hline x_i x_j \quad x_i \overline{x_j} \end{array}$	$x_i x_j + x_i = 2 x_i x_j - x_i x_j + x_i =$ $= 2 x_i x_j + x_i (1 - x_j) =$ $= 2 x_i x_j + x_i \overline{x_j}$

So, what are that like if I have to add this x ij with x ji, so, that will be; that means, as this becomes x ij plus x ij or x ji or x ij both are same. So, that will be mentioned as 2 x i of j x j x i and x j. So, that can be written as the x i and x j that can be written on the next MSB position or that will be shifted to the next MSB position. Just like the same what I said just now that x i if I want to multiply with x i that will be nothing but my x i.

Now, suppose I have to add this particular terms; $x_i \times x_j$ plus x_i . So, can I reduce this term? Yes I can reduce or I can write these particular terms in a different way. How? If I write this $x_i \times x_j$ plus $2x_i$ in this manner that is $2x_i \times x_j$ minus $x_i \times x_j$ plus x_i ; that means, $1x_i \times x_j$ has been minus from this $2x_i \times x_j$. So, $2x_i \times x_j$ plus x_i is common over here. So, I take it common and it can be written as 1 minus x_j so, 1 minus x_j is nothing but \bar{x}_j .

So, that means, now this $x_i \times x_j$ plus x_i that can be written as $x_i \times \bar{x}_j$ at this particular position and as this is $2x_i \times x_j$; so that means, this will be shifted to the next MSB position. So, that is why this $x_i \times x_j$ has been move to the next MSB position ok, got it? So that means, this particular terms $x_i \times x_j$ plus x_i that can be written as now $x_i \times x_j$ in the next MSB side whereas, this $x_i \times \bar{x}_j$ on the corresponding whatever; that means, position we are trying to add this $x_i \times x_j$ plus x_i . So, on that particular position this can be written.

So that means, this kind of optimization we can do whenever we are doing this squaring operation ok. So, that means, now at I am removing some of the gates using this kind of this logical optimization, so that means, now I can save more number of gates. So, saving more number of gate means I can reduce the complexity, logical complexity and reducing the logical complexity which we will give me in terms of savings in terms of power consumption as well as sometimes the improvement in the speed too. So, this is not the fact that all the time I will get the benefit in terms of speed, I may not get, but here I can get in terms of improvement in terms of speed.

(Refer Slide Time: 09:38)

Squaring Using Lookup Tables

input= a	output= a^2
0	0
1	1
2	4
3	9
4	16
...	...
i	i^2
...	...
$2^k - 1$	$(2^k - 1)^2$

for relatively small values k
 2^k words $2k$ -bit each

So, then there is another method of doing this squaring. So, this is the; that means, generic method for multiplication which we have already followed for doing this

squaring operation. So, another thing also we can do that means, that is as the number is basically fixed; number is basically fixed means their number is same. So that means, a number is same means the number can be either 0 1 2 3 4 5 6 7.

So, that means, any decimal number and squaring means, it will be just 4 will be multiplied with 4 only ok. So, what we can do? We can store this particular squared values in the ROM or in a memory ok, where you can see that this 0-th location contains the values of 0. This one location of the memory contains the values of 1, then second location of the memory contains the values of 4, then 9, then 16; that means, here this is the address values.

This is the address values means, so, this is the 0-th location, this is 1, this is 2, 2nd location, this is 3rd location, this is 4th location, something like this. If I want to choose that means, this is this for k bit if I that means, if the ROM size is of; that means, k bit ok. So that means, now I can sorry this input is of k bit, so that means, now I can form 2 to the power k minus 1 number, I can get. So, and this corresponding this ROM can store the values of 2 to the power k minus 1 square values.

So that means, if I choose this k bit for the input, so that means, the length of the corresponding ROM size will be 2 to the power k minus 1. Not only 2 to the power k minus 1, I need this will be basically 2 dimensional; this ROM will be 2 dimensional. How it will be 2 dimensional? Because, to store this 1, 4, 9, 16; this value, I need the numbers will be represented or stored in the ROM in terms of binary.

So, that means, for one of this location this value contains the location value contains let us say 1 and but this corresponds this particular value that may be contains of 16 bit ok. So that means, if this is the; if this is the; that means, 2 to the power k minus 1, so, twice k bit I require to store this particular 2 to the power k minus 1 square values. Why? Because, if you see that just see if for considering this 2 ok, so, they if you consider 2, so that means, now I require the number is basically squared; that means, that is 4 ok.

So, that means, to if I consider k number of bit for this I need 2 to the power k minus 1, sorry 2 to the power k number of positions or the length of this ROM will be 2 to the power k along with that each of the length of this twice of k bit each.

So, that means, now if my k is more, if I consider the k is of that means, in a wide range, so, at that time the corresponding ROM size will be very much high. So, very much high means, it will consume more power, more area as well as more power. So, that is why this particular this look up table base or this is ROM based or I can say this is as lookup table based; lookup table means, where we store the values. So, this lookup table based this squaring circuit it is useful whenever the values of k is relatively small.

For higher values of k, we can that means, instead of getting the; that means, here what is the advantage? That means, we do not a need any computation; that means, the complexity of the circuit that will be reduced because it is a stored. So, I will face the data from the memory only. So, if the corresponding suppose the input is let us say let us consider 3, so, automatically the 9 will be retrieve from the memory.

So, the complexity wise it is very much simpler in compared to the previous circuit, but the thing is that as the k value will be more, so that means, I require more area as well as more power. At that time the advantage, though the complexity will be lesser but the consumption of area and power that will be on the higher side. So, on that particular case it is not; that means, suggested to use this kind of architecture for squaring operation.

(Refer Slide Time: 15:37)

Multiplication Using Squaring

$$a \cdot x = \frac{(a+x)^2 - (a-x)^2}{4}$$

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

So, then how we can do this multiplication using this squaring? So, how we can do this multiplication using squaring? Suppose, I need to multiply a into x, how we can do? We know the fact that actually this is the basic arithmetic we have learnt in our high school days.

So, that is a plus x is whole square minus a minus x is whole square divide by 4 so, that I will get a x. Why? Because here what I will get; a plus x said a plus x of whole square means, a square plus 2 a x plus x square minus of a square plus sorry minus 2 a x plus x square. So, minus a square minus x square that will be that means, nullify from here. So, 2 a x I will get here and 2 a x I will get from this particular things.

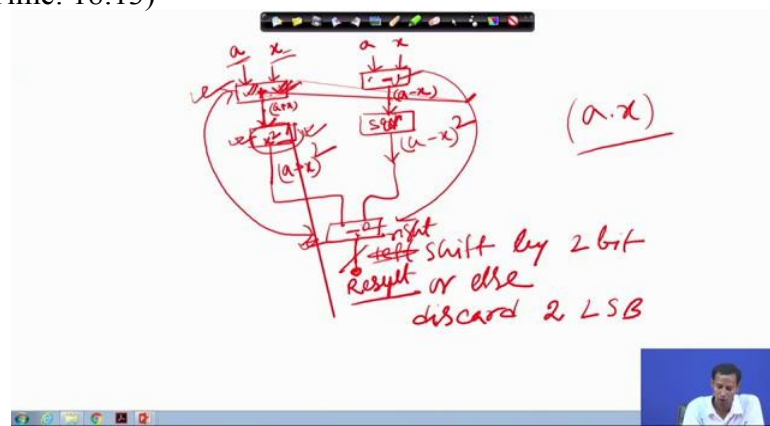
So, 2 a x 2 a x 4 a x divided by 4 means that is nothing but ax. So, division of 4, means, if I just discard the last 2 LSB position, so, easily I will get this divide by 4. So, then what I have to do here circuit wise or the architecture wise what will be the architecture? Suppose, I am having this squaring circuit, then initially what I have to do? Suppose, this 2 terms; that means, the multiplier and the multiplicand they are a and x, so, in one particular circuit, so, if I just draw this corresponding circuit, so at that time how it will be? It will be just like.

(Refer Slide Time: 17:36)

$$\begin{aligned}
 a \cdot x &= \frac{(a+x)^2 - (a-x)^2}{4} \\
 &= \frac{a^2 + 2ax + x^2 - a^2 + 2ax - x^2}{4} \\
 &= \frac{4ax}{4} = ax
 \end{aligned}$$

So, a into x if I have to do, so that will be minus a minus x whole square divide by 4. So, means this is a square plus 2 a x plus x square minus a square plus 2 a x minus x square. So, these and these this will be nullified divided by 4 mean 4 of a x divided by 4 so, that is nothing but my a x. So, if I just want to draw this particular circuit, so, at that time what will be its architecture?

(Refer Slide Time: 18:13)



It will be just like, so, if this is a , this is x ok. So, in one particular case I have to generate a plus x . So, then again I will have another particular circuit which is minus x .

So, this will give me a minus x . So, now, I will have this corresponding squaring circuit. Suppose, this is the x square circuit, so, this I will get. So, from here what I will get? That is a plus x of whole square. So, again if I just pass this along with another sorry this is just squaring circuit.

So, if I just pass this value then again I will get a minus x of whole square ok. Then again what I need? I need 1 subtraction of this ok. And from here what I have to do? I have to left shift the results of this, sorry yes left shift sorry, this is right shift by 2 bit or else what I can write?.

Discard 2 LSB from this particular results ok. So, if you do that then I will get a into x ok. So, complexity wise or if I just try to calculate the corresponding delay for that so, then if you see that here I am getting the; that means, for the squaring circuit I can optimize it ok. But here I need 2 additional, here I need a 1 additional adder, here I need a 1 additional subtractor.

If I just say in this particular path, I need 1 subtractor here additional, 1 subtractor here additional. So that means, by using this particular method, if I want to calculate this that means, the multiplication, so, at that time I can get or I may get for if I consider more number of bits for this a and x , so, at that time this particular architecture will not give me that much benefit. Why?

Because, whatever benefit I will get in the squaring circuit that will be; if this particular bit that means, if the length of a and x they are; that means, more, so, at that time the complexity or there area requirement or the speed requirement for this extra overhead of this adder and subtractor that will be degrade the corresponding optimization whatever I am getting for this squaring circuit.

(Refer Slide Time: 21:55)

Multiplication Using Squaring

$$a \cdot x = \frac{(a+x)^2 - (a-x)^2}{4}$$

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

So, that is why this particular architecture ok, this particular structure is beneficial whenever the corresponding length for this a and x, the bit of a and x they are less. Whenever they are more at that time this particular architecture is not that much beneficial ok. So that means, hardware design of that if that is not at all advantageous at that time. So, this is the another method of doing the multiplication using squaring method, so that you can use in your system. So, apart from that there are again this multiplication architecture is such a; that means, dynamic field. People are doing more research on that for finding out one efficient architecture for designing the multiplication.

So, if you find that means literature you can get there are some Vedic multiplication basic Vedic multiplication methods are there in the; that means, in the ancient Vedic era. So, that particular algorithms, now you can map into the hardware or you can design into the hardware to get the multiplier architecture which will be beneficial in terms of speed or in terms of area or in terms of power; different aspects you can chose for ok.

Apart from that again there are this based on this approximate computing also some of the multiplier has been designed, means what? In a approximate computing what we do? We intentionally we put some of the error or we just discard some of the bits and then we do this multiplication and we pre calculate those errors. So, what will be that means, intentionally as I am skipping some of the bits.

So, from the beginning we know that because of this skipping of the bits so, what will be the maximum error, what will be the quantity of the error I can accumulate during that whole multiplication operation. So, at the end whenever we will produce the result, so, at that time we will try to calibrate with those errors; that means, pre computed errors to get the proper result.

So that means, as I have reduce the number of bits from the beginning; that means, the complexity of the hardware at that time it will be lesser ok. But apart from the at that time the at the same time what I have to remember that the results that should be correct.

So that means, for the corresponding approximation what will be the error so, that will be again added at the final stage to get the accuracy also at the topper level or in a adequate level. So, that type of multiplier architectures is also available in the literature, people have worked on that and that is based on this approximate computing this multiplier design.

So, that type of architecture we are not discussing in this particular class. So, if you need then I will be just putting it, just as a supplementary material in the discussion forum that I will be supplied ok. Then, till now what we have seen that the multiplier or the multiplication operation in the considering the number in the like in the integer part.

So that means, here what we have seen? The number that are like 1 2 3 4 10 11 22; so, those the numbers are basically integer one. Now, suppose if you see most of the DSP system; in most of the DSP systems, the number DSP system means, suppose if you want to do the DFT or if you want to do the filtering or if you want to do the FFT or if you want to do the any transform, like Hilbert transform, cosine transform or then Hartley transform, sorry.

So, this kind of transform they require the coefficient values or in real time DSP system the coefficient values of this filters or this transforms are basically considering the fact they are in the decimal or the fractional part. That means, mostly they are dominated by the fractional part. If you just see that whenever we are doing the multiplication, we do not know what will be the value for multiplier and multiplicand so, at that time the architecture is different. Whenever we know that the 2 numbers are same, so, at that time we can do more optimization.

Now, another thing is that in if FIR filter most of the; that means, the multiplication which is being used in the DSP system, there the values are constant. That means, suppose if I say that this 3-tap filter with this kind of specification, the corresponding filters parameters or the coefficient values they are fixed for that particular specification ok.

The same things happen suppose this let us say 8 point FFT if I ask you. So, the 8 point FFT, the corresponding twiddle factors they are basically fixed, the values of that that will be fixed. So, considering these fact, whenever we need the multiplication operation on those system, so at that time can I do it more optimization? That means, the structure can I do it more optimized? Yes, I can. How I can?

Because the number is constant means, I know the corresponding bit values of each of these particular coefficients. So, considering this fact that now I can optimize the multiplier in a better way ok. So, I do not require to use any generic multiplication at that time to do this multiplication on the DSP system.

So, if I can use if I use it, so at that time intentionally or unintentionally I will use more of the area or more of the power though because, I have the provision to do the optimization because I know the numbers from the beginning as a number is already fixed. So, I can do the optimization or I can play with that. So, how I can do or what is the; that means, at that time how this technique will be or what will be the techniques to do this optimization that we will see in the next class.

So, thank you for today's class. So, next day we will see this constant multiplication and then not only constant multiplication this reconfigurable constant multiplication how we use and where it is being used, how we can that means, come to the optimize circuit or how we can design it that we will see in the next class onwards.

So, thank you for today.