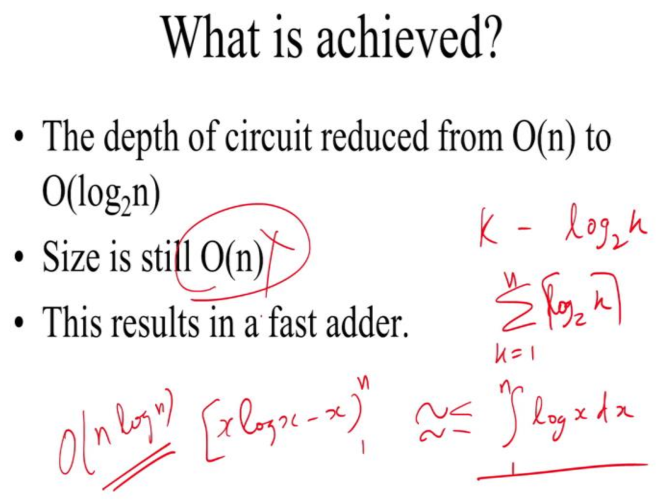
Computer Organization and Architecture

Prof. V. Kamakoti
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture – 05
Fast Multiplier Circuit

Good morning. So, I hope you are able to see these just go through your notes with respect to you know construction of a carry look ahead adder that we saw in last class, and now you are in a position to appreciate how carry look ahead adder can do addition up to n bit numbers and log n time, well a carry look ahead adder take n units of time right.

(Refer Slide Time: 00:42)



What is achieved?

- The depth of circuit reduced from O(n) to O($\log_2 n$)
- Size is still O(n)
- This results in a fast adder.

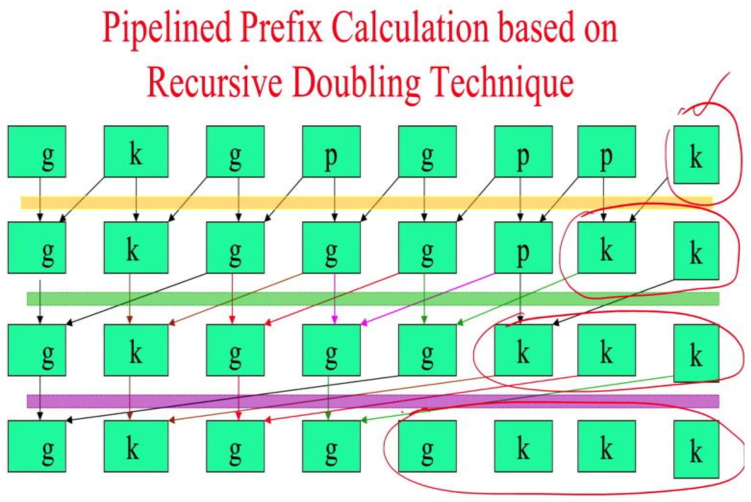
$k - \log_2 k$

$\sum_{k=1}^{n} \lceil \log_2 k \rceil$

$O(n \log n)$ $\quad [x \log x - x]_1^n \quad \approx \leq \int_1^n \log x \, dx$

And we stopped with what is the total area required is it order n or order n log n. So, did somebody look into it I said the size is still order n; is this a correct statement or a wrong statement?

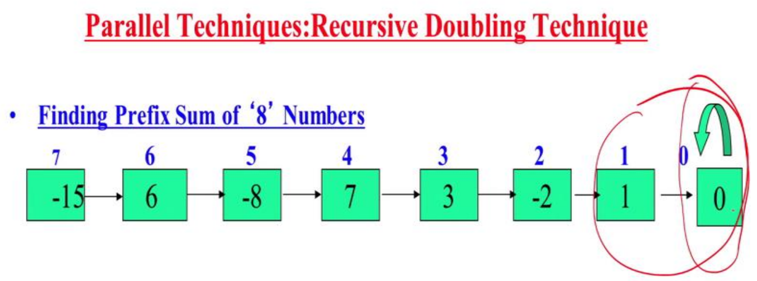
(Refer Slide Time: 00:56)


Pipelined Prefix Calculation based on Recursive Doubling Technique

Please note that the 0 th stage or a 0 right is useless after does not do any computing at all this.

So, this stops here there is no need for computing it beyond this point, for the first stage there is no need for computing beyond. So, these two are empting the after this first stage or second stage whatever you call, and in these 3 are empty here and 4 are empty right.

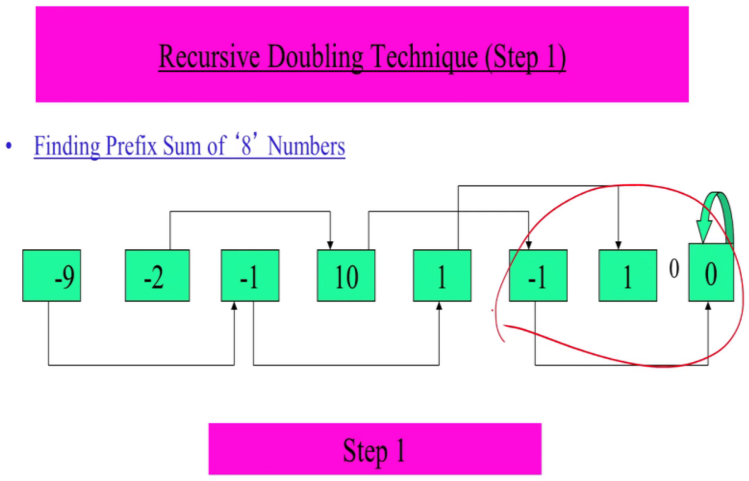
(Refer Slide Time: 01:42)


Parallel Techniques:Recursive Doubling Technique

- Finding Prefix Sum of '8' Numbers

So, please note that if you go back to the previous slide, in the 0 th stage only 0 points to 0 and 1 has reached this; the moment you start pointing to this then after that you do not need this correct right.

(Refer Slide Time: 01:58)



Recursive Doubling Technique (Step 1)

- Finding Prefix Sum of '8' Numbers

At the end of first stage this is already 2 two entries are pointing to this. So, if I take the n th element.

So, how do you analyze this size? If I take the n th element or if I take the k th element when will it reach this? Say every time is going to point to some entry which is twice as far as what it was pointing earlier for example, this was pointing one before it in the next stage it will point two before it, the next stage it will stop point 4 before it, and it actually becomes useless for computing meaning, there is no constructive computing necessary after its starts pointing to the last element correct. So, every time I double the distance to which I point two; and if I point to the last element then after that I there is no computation left for me. So, how much time will I take to point to the last element or how many steps will I take to point to the last element for.

Student: (Refer Time: 03:05).

Log of if I am the k th element, how many steps will I require to go and reach the first element every time I am reaching double the distance, double the distance. So, first time pointing to 1 before me next 2, next 4, next 8 like that how much time will I take to reach k?

Student: (Refer Time: 03:29).

Log k right. So, I need log k computation. So, I need to be repeated for log k times right after that it is just that value can keep propagating. So, the circuit actually I need a circuit for computation of that star operator correct? That circuit needs to be replicated log k times for a for the k th element. So, how many such replications I need can I now write it as a formula? So, for the k th element I need log k replications at least in log k steps I need that circuit repeat right. So, what is that total number of replications note that this is a same star operation so what is the total number of replications sigma k equal to 1 to n log k, some seal we can put. So, this can be approximated to can we do this sum is right. So, this will be lesser than lesser than or equal to this, because we are only interested in discrete points while this is a continuous function. So, how do you integrate log x d x.

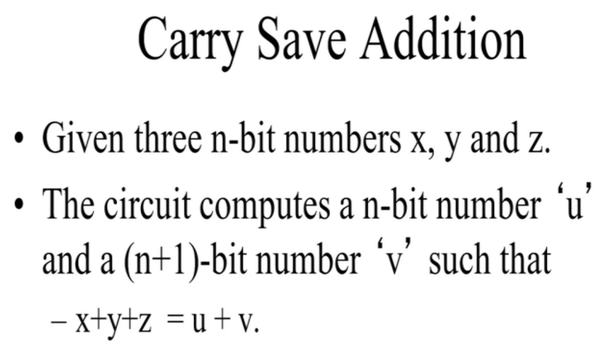
Student: (Refer Time: 04:49). X log x minus x.

X log x.

Student: (Refer Time: 04:58).

And this is go from 1 to n. So, this would be order n log n, log n minus n that is 5 or n into log n minus 1 this will be still order n log. So, this is wrong fair enough are you able to follow this good. So, I want you to find see I have voluntarily introduced errors in my slides, I want you to find out where the errors are and I may ask you a quick question on see I do not take attendance, but the way I ensure attendance is that I will ask one quick question on you know what are the errors are in my slide, right.

(Refer Slide Time: 05:56)

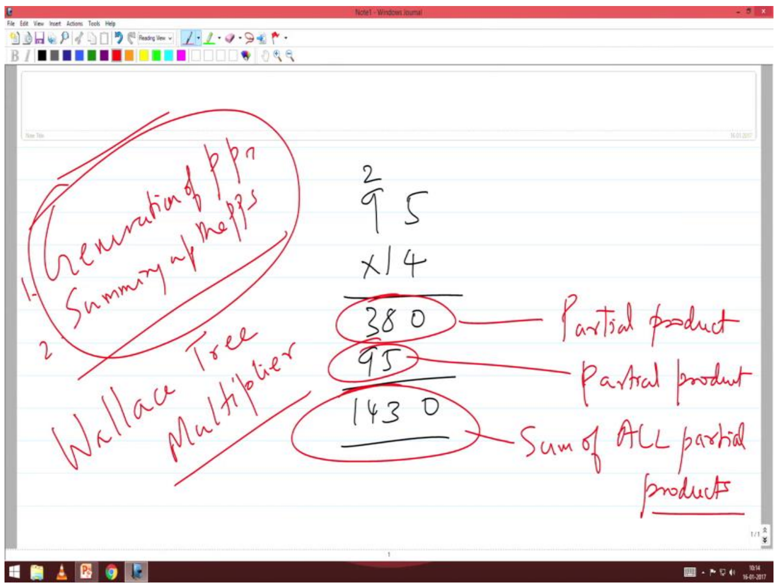


Carry Save Addition

- Given three n-bit numbers x, y and z.
- The circuit computes a n-bit number ‘u’ and a (n+1)-bit number ‘v’ such that
  - $x+y+z = u + v$.

Now, we will go in to the next important operations in computing namely multiplication.

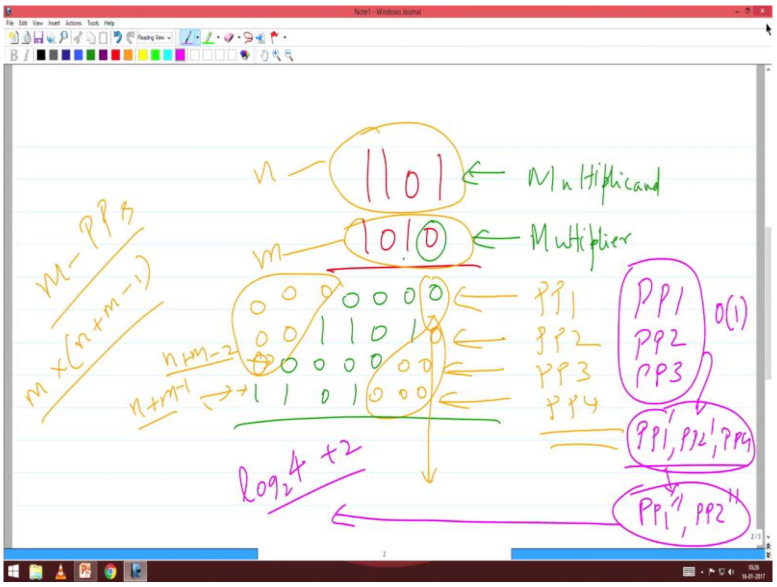
(Refer Slide Time: 06:07)



Now how do we multiply? So, let us go here suppose I want to multiply 95 into 4, 95 into 14. So, I just say 4 plus 4 into 524 into 936, 380 correct and the 95 then I sum it fine. So, that right. So, each one of this is a partial product, when you actually sum of the sum of all partial products. So, high school multiplication involves two steps one is generation of partial product, and two summing up re partial products fine are you able to understand this. Now entire multiplication should concentrate on how effectively we can do this or how quickly we can do this fine.

So, there are several algorithms that are reported let us not bother so much for, we will introduce a latest algorithm. So, what will be teaching in this class is called a Wallace tree multiplier fine. So, please understand that I need to generate partial products and I need to sum of these partial products, and please note that this is not decimal arithmetic we are going to handle it in binary.

(Refer Slide Time: 08:22)



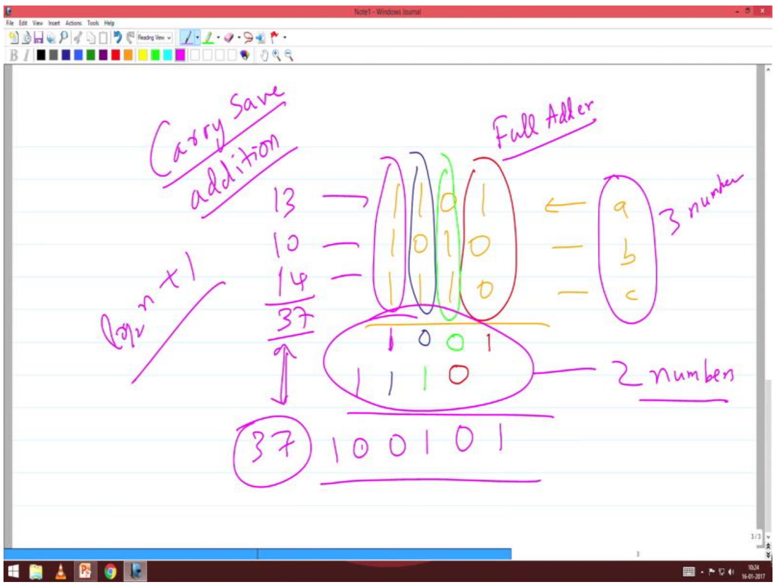
So, let us say I want to multiply right. So, what are the partial products here I take this 0, I take this 0 and it with 1 1 0 1. So, 0 0 0 0, 1 1 0 1, 0 0 0 0, 1 1 0 1 correct and we add this. So, the generation of the partial products is basically to take every bit of one of the so every bit of the multiplier.

Let me call this as multiplicand let me call this as multiplier. So, we take every bit of a multiplier and multiply with the every bit of the multiplier and multiply with the multiplicand are ANDed with their multiplier. So, we do what we call as a every bit of the multiplicand get and with one of the bits of the multiplier next this next this. So, this is how. So, these are all the partial products that we see here, let me call it P P 1, P P 2, P P 3 and P P 4. So, if the multiplicand is n bits and your multiplier is m bits we get m partial products correct we will get m partial products, and with shifts right it will keep shifting right. So, the last one would be shifted by m bits.

So, it will be this would be n plus m bits because this is n bit plus n plus m minus 1 bits n plus m minus 2 bits and so on right. So, this is effectively these are all zeros, we can still fill up. So, I could say that we have now m numbers each of n plus m minus 1 bits also. So, make this whole thing look easy right, but these are all always zeros these are all always zeros; when we actually construct a circuit please note that we will not try to have you know some elements for these right some circuits some circuit elements for these because they are already zeros. So, we this is just this is the answer itself first bit of your answer is generated the moment you generate the partial product.

So, fine are you able to follow this, now any doubts in this right. So, I generate partial products and I sum them the way I generate partial product in binary arithmetic is to take up bit and it with the bits of the other operator and keep doing it every bit of the multiplier, and then sum it up with necessary left shifts fine able to follow? Now suppose let us say in partial product I get n or m partial products and I am adding those m numbers. So, how can we add two numbers?

(Refer Slide Time: 12:19)



See so for example, suppose I say. So, how can I add 3 numbers let us go and do this suppose I would not add 1 1 0 1 with 1 0 1 0 with 1 1 1 0.

Suppose I want that is right for each of. So, these are 3 numbers a b and c, the way I can add it is take each of this in parallel the quickest way of adding it is let us take this in parallel, each of this columns in parallel right let me use a full add up circuit with 3 inputs it will generate me a sum bit and a carry bit. So, this will generate me a sum bit as 1 and carry bit as 0, I am writing the carry bit here this will generate me a sum bit of 0 and a carry bit 1, this will generate me a sum bit of 0 and a carry bit of 1 and this will generate me a sum bit of 1 and a carry bit of 1 fine.

Now, you add these two this will be 1 0 1 correct fine right. So, what is this this is 13, 10 and 14. So, this is 27, 37. So, this is 32 plus 5 37 both are equal. So, what did I do? I converted the problem of adding 3 numbers to the problem of adding two numbers; because I have an adder which can add only two numbers now what I have done I have converted the problem of adding 3 numbers to the problem of adding two numbers, and how much time does it require for me? One unit of time because I use a full adder circuit for this correct I use a full adder circuit for this.

So, I use 4 full adders in parallel concurrently and immediately generate to two numbers from 3 numbers, and those two numbers I can use any carry look ahead adder and add it right. So, assuming that the carry look ahead adder will take log n time, so adding 3 numbers can also be achieved in log n plus one time that one is full order delay. So, is right. So, this type of an addition is called as carry save addition right. So, every time what I did, I generate I took each of these columns in parallel, generated as sum bit and the carry bit and I just arrange the carry bit with one shift to the left and added the sum bits to that. So, I save the carry bit for adding at a later stage, in this stage I get the sum bit and I also a carry bit which I say and use it as a later stage for finding the result right.

So, this is what we term as carry shift addition right any doubts in this. So, the problem of adding 3 numbers get translated to the problem of adding two numbers, by this carry save addition, and this will be using for our multiplying right. Now how can we do this here? So, I actually have 4 numbers to be added, I can take the first P P 1, P P 2 and P P 3 right and I can generate what? I will generate two numbers. So, let me say P P 1 dash and P P 2 dash; this P P 1 dash and P P 2 dash I will take, and add it with P P 4 to generate another two numbers namely P P 1 double dash and P P 2 double dash and these two I can use a carry look ahead adder and. So, the problem of adding 4 numbers right.

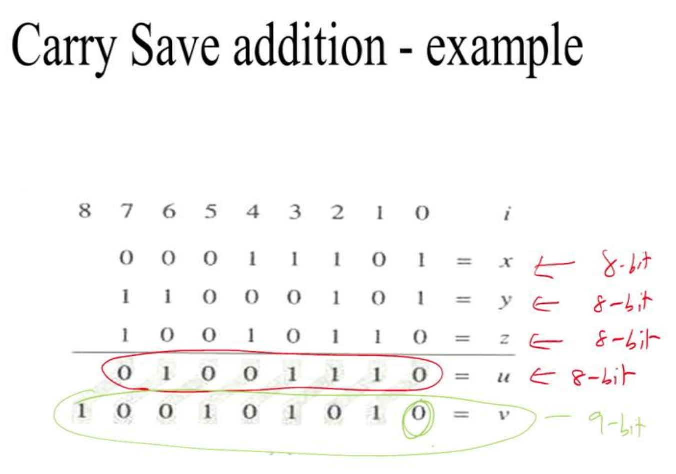
I can just in constant time I can convert this 3 numbers into 2 numbers, again in constant time this will take order one time one full adder, and again in constant time I can convert this new there are two new numbers that are generated along with the old P P 4 that I can convert it to two times. So, in log n plus two steps log n for this addition, this is again I can use a carry look ahead adder plus two steps I can finish off multiplication of these two numbers not log n plus two in this case this is 4, 4 because we have n is 4 right.

Now this is the basis and now we will extended for I will just shown it for multiplying to 4 by 4, 24 bit numbers now I will extend it for multiply a 2 n bit numbers right and this is the basis of Wallace tree multiplication; fair, fair enough are you able to follow did you all understand what is carry save addition Palakkad.

Student: Yes sir.

And how the carry save addition could be used in the context of multiplication right I just gave this example here, these two are the things that I want you to understand here fine? Now we will go back to I will formally tell you how carry save addition works right. So, given 3 n bit numbers x y and z, the circuit computes a n bit number U and a n plus 1 bit number V such that x plus y plus z equal to U plus V in that is what we have seen here.

(Refer Slide Time: 19:20)



Carry Save addition - example

8	7	6	5	4	3	2	1	0		i	
	0	0	0	1	1	1	0	1	=	x	← 8-bit
	1	1	0	0	0	1	0	1	=	y	← 8-bit
	1	0	0	1	0	1	1	0	=	z	← 8-bit
	0	1	0	0	1	1	1	0	=	u	← 8-bit
1	0	0	1	0	1	0	1	0	=	v	— 9-bit

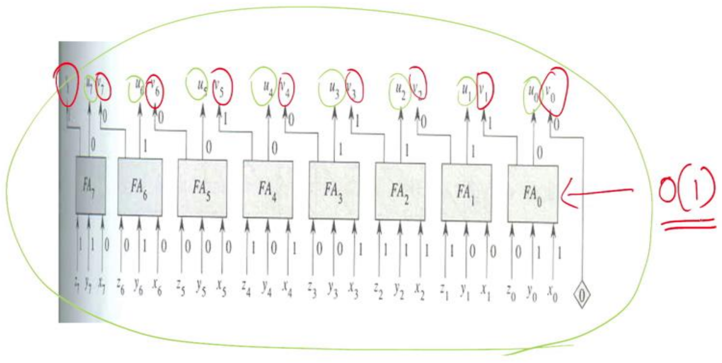
So, let us see how that is. So, this is x this is an n bit number 8 bit number, and this is y which is also another 8 bit number this is z which is also another 8 bit number.

Now, I have generated another 8 bit number which is U this is very important for you to note this U is 8 bits and the v, V is actually 9 bits. Please note that this 0 is just applauded I shift the carry by 1 bit to the left as you have seen in the carry save addition. So, formally stating if I want to add 3 n bit numbers that problem could be converted in constant time to a problem of adding 1 n bit and another n plus 1 bit number correct? Adding 3 n bit numbers in constant time can be converted to a problem of adding 1 n bit and 1 n plus 1 bit numbers.

Please note that n and n plus are very important to note here because finally, this is what we will use to construct the final circuit this is fine any doubts? 2 n bit numbers 3 n bit numbers become addition of 3 n bit numbers become addition of 1 n bit and 1 n plus 1 bit number, in constant time using n full adders is this is this clear yes or no; yes ok.

(Refer Slide Time: 20:59)

Carry Save Adder Circuit



FA_7 FA_6 FA_5 FA_4 FA_3 FA_2 FA_1 FA_0
u_7 v_7 u_6 v_6 u_5 v_5 u_4 v_4 u_3 v_3 u_2 v_2 u_1 v_1 u_0 v_0
z_7 y_7 x_7 z_6 y_6 x_6 z_5 y_5 x_5 z_4 y_4 x_4 z_3 y_3 x_3 z_2 y_2 x_2 z_1 y_1 x_1 z_0 y_0 x_0
0
O(1)

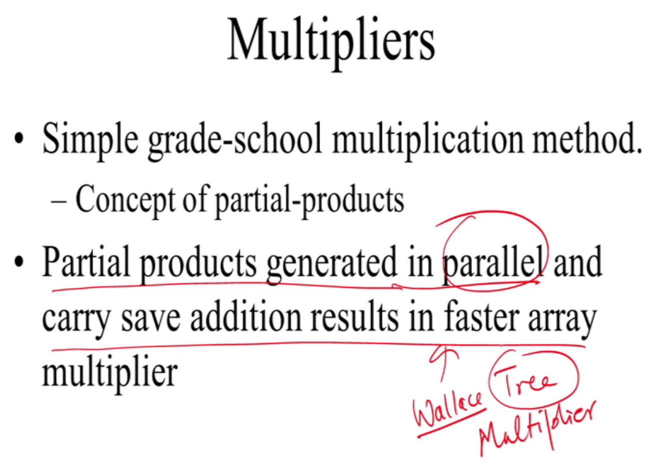
So, this is the basic circuit as you see here you will have n full adders, and each full adder will generate a sum bit which we call it as U 0, U 1, U 2, U 3, U 4, U 5, U 6, U 7 and each full adder will also generate carry bit which we call which is V 0 is appended 0 V 1 is V generated by this V 2, V 3, V 4, V 5, V 6 and V 7 and V 8. So, U is n bit number and V is a n plus 1 bit number yes.

Student: Input of the full adder (Refer Time: 21:46) only V or. 3 pin full adder or

Full adder means it is a 3 bit full adder is a module type, just as an x r circuit and 3 n gate an on r gate it will generate sum of 3 bits and the carry of this 3 bits right fair enough.

So, this is a construction. So, please this is I am showing this slide to show that this entire operation will take constant time or order one time one is constant ok.

(Refer Slide Time: 22:16)



Multipliers

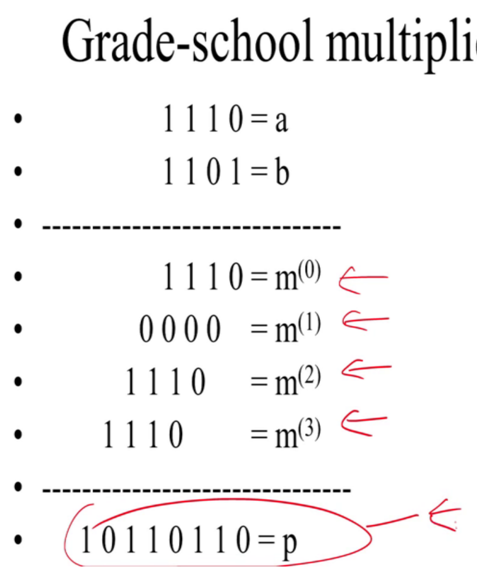
- Simple grade-school multiplication method.
  - Concept of partial-products
- Partial products generated in parallel and carry save addition results in faster array multiplier

Wallace
Tree
Multiplier

So, we will very quickly go into the multiplier. So, this is simple grade school multiplication method as I explained, we first generate partial products and then in parallel please note that this is done in parallel, and then we do carry save addition which will give us this faster array multiplier what we call this faster array multiplier was due to Wallace, and so we call it as Wallace tree multiplier. Wallace is the name of the scientist.

But why tree we will just see very quickly why it is ok.

(Refer Slide Time: 22:54)

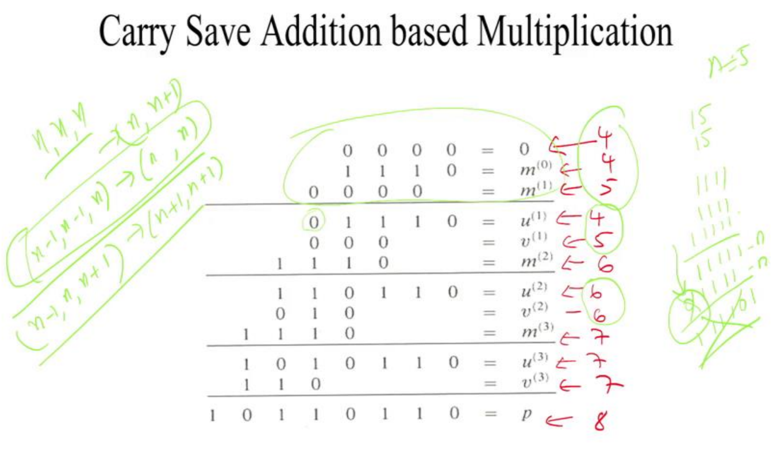


Grade-school multiplication

$$\begin{array}{lrl}
\bullet & 1\,1\,1\,0 & = a \\
\bullet & 1\,1\,0\,1 & = b \\
\bullet & \text{------------------------------} & \\
\bullet & 1\,1\,1\,0 & = m^{(0)} \leftarrow \\
\bullet & 0\,0\,0\,0 & = m^{(1)} \leftarrow \\
\bullet & 1\,1\,1\,0 & = m^{(2)} \leftarrow \\
\bullet & 1\,1\,1\,0 & = m^{(3)} \leftarrow \\
\bullet & \text{-------------------------------} & \\
\bullet & 1\,0\,1\,1\,0\,1\,1\,0 & = p \leftarrow
\end{array}$$

This is the same thing that I worked out 1 1 1 0 with 1 1 0 1 this is what we need to add, and please note that we have generated 4 partial products m 0, m 1, m 2, m 3 we add these 4 partial products to give me the final answer which is. So, how can we go and do this.

(Refer Slide Time: 23:13)



Carry Save Addition based Multiplication

$$\begin{array}{cccccccccll}
 & & & 0 & 0 & 0 & 0 & = & 0 & \leftarrow 4 \\
 & & & & 1 & 1 & 1 & 0 & = & m^{(0)} & \leftarrow 4 \\
 & & & 0 & 0 & 0 & 0 & & = & m^{(1)} & \leftarrow 5 \\
\hline
 & & & 0 & 1 & 1 & 1 & 0 & = & u^{(1)} & \leftarrow 4 \\
 & & & 0 & 0 & 0 & & & = & v^{(1)} & \leftarrow 5 \\
 & & 1 & 1 & 1 & 0 & & & = & m^{(2)} & \leftarrow 6 \\
\hline
 & & 1 & 1 & 0 & 1 & 1 & 0 & = & u^{(2)} & \leftarrow 6 \\
 & & 0 & 1 & 0 & & & & = & v^{(2)} & - 6 \\
 & 1 & 1 & 1 & 0 & & & & = & m^{(3)} & \leftarrow 7 \\
\hline
 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & = & u^{(3)} & \leftarrow 7 \\
 & 1 & 1 & 0 & & & & & = & v^{(3)} & \leftarrow 7 \\
\hline
1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & = & p & \leftarrow 8
\end{array}$$

n, n, n → (n, n+1)
(n-1, n-1, n) → (n, n)
(n-1, n, n+1) → (n+1, n+1)
n=5
15
15

So, first what we will do is that we will first add. So, m 0, m 1, m 2, m 3 we can do it as follows right we just add m 0, m 1. So, these are very new way of doing it we first add m 0, m 1 and 0 right we get u 1 and v 1. So, we add please note that we this is a dummy variable this is a dummy entry and these two are partial products each of 4 and 5 bits right.

So, I add 3 numbers one of 4 bit, another f 4 bit and another of 5 bit and I get two numbers one is actually a 4 bit number, another is actually again a 5 bit number please note that if each are 5 5 bits I get 5 and 6, but since both are two of them are 4 and another is a 5 bit you get an answer which is a 4 bit and a 5 bit number. Now to this 4 bit and 5 bit I add another number m 2 which is actually 6 bits right. So, I get a 6 bit number and another 6 bit number again 4 5 6 I get 6 bit numbers, to which I add another number which is 7 bits. So, I get one two 7 bit numbers which basically gives 8. So, this is a sequential way of doing this addition right. So, what we why I am put this slide here is to basically explain you how things can work here.

So, I have 4 partial products to basically start with, I have 4 partial products and I want to add this method gives you sequential way of adding it. So, I start with a dummy 0 m 0 m 1 so 3 numbers that gets converted into 2 numbers right? But what I want you to note here is the most important thing that I want you to note here is that this 4 4 and 5 gives me 4 and 5. 4 4 and 5 will give me only two numbers which are 4 and 5 bits let as I told you if I have 3 n bit numbers it will give me an n bit and n plus 1 bit number.

But if I have say n minus 1, n minus 1 and n this will give me only n minus 1 and n bit right or in sense n, n and n this is fine. Now I get a 4 bit number and a 5 bit number and a 6 bit number, this is going to give me two 6 bit numbers. So, if I have n minus 1, n and n plus 1 I get n plus 1 comma n plus 1 yes ok.

So, very simple if I have 4 bit number what is the maximum value I have? 15 another would be 15 right and. So, this should be 1 1 1 1, 1 1 1 1 and if I have a 5 bit here right whatever be that 5 bit that is u 1 take that.

Student: (Refer Time: 26:59) u 1 is 5 bit number.

No, here 4, 4 and 5 first.

Student: In the second side using u 1 (Refer Time: 27:08).

Yeah. So, n is 5.

Student: (Refer Time: 27:15) not 4 5 6 (Refer Time: 27:15).

No this is no, but this is 0 this is 0.

Student: (Refer Time: 27:21).

Yeah, but in sense it is 4, see when you are doing this addition we are seeing a 4 bit number here that is why I am saying. So, I if I have, let us just prove this formula right. So, if I have n n and n plus 1 right n and n minus 1, n minus 1 and n plus 1 n. So, n is 5 here. So, I put even 1 1 1 1 1 1. So, what I will get here 1 1 1 1 0, the sum bit sorry 1 1 1 and 1 sorry sorry. So, this gives me a carry, carry and this gives me 0 ok.

So, what I will get here? I will get 1 right. So, if I have n n n minus 1 n minus 1 n oh I get 6, yeah n minus 1 n minus 1 n wait n minus 1, n minus 1 n no I am not doing the addition right, n minus 1, n minus 1, n this is n and yeah. So, if I have n minus 1, n minus 1, and n the worst I will get this n and n because this is always 0 form. So, this should be fine. So, if I have n minus 1, n minus 1, n, I get n and n is it fine no ha what should I get.

Student: Actually if we have 4 4 and 5 (Refer Time: 30:00).

What will I get?

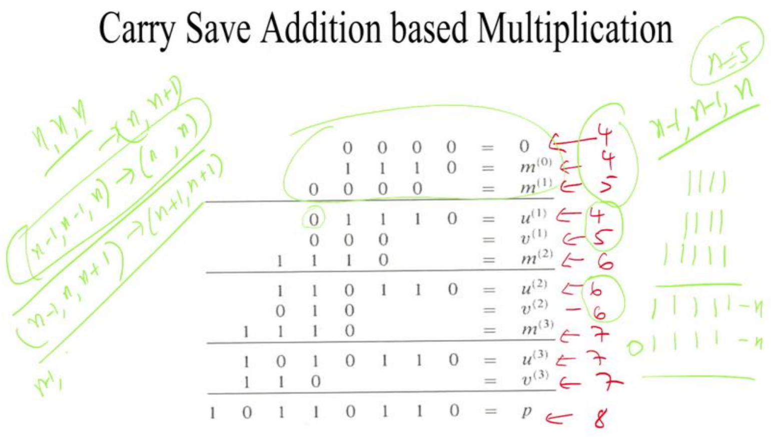
Student: (Refer Time: 30:01) some bit should be 5 right. So, (Refer Time: 30:02) 0 1 1 1 1.

5 that is what I am saying n is 5 here know, sum is 5 and carries also 5.

Student: Carry bits should be (Refer Time: 30:11).

Why carry bit cannot need not be 6 because the carry will be 0 for sure, this carry will always be 0. See I put 15, 15 and 31 the maximum I added even further the carry will always be 0 can you work it out again.

(Refer Slide Time: 30:40)



Carry Save Addition based Multiplication

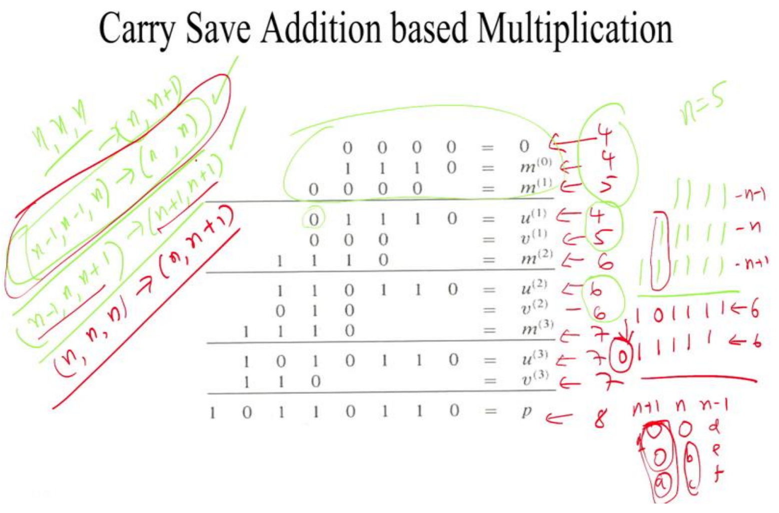
$$
\begin{array}{cccccccccl}
 & & & & 0 & 0 & 0 & 0 & = & 0 \\
 & & & & 1 & 1 & 1 & 0 & = & m^{(0)} \\
 & & & 0 & 0 & 0 & 0 & & = & m^{(1)} \\
\hline
 & & & 0 & 1 & 1 & 1 & 0 & = & u^{(1)} \\
 & & & 0 & 0 & 0 & & & = & v^{(1)} \\
 & & 1 & 1 & 1 & 0 & & & = & m^{(2)} \\
\hline
 & & 1 & 1 & 0 & 1 & 1 & 0 & = & u^{(2)} \\
 & & 0 & 1 & 0 & & & & = & v^{(2)} \\
 & 1 & 1 & 1 & 0 & & & & = & m^{(3)} \\
\hline
 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & = & u^{(3)} \\
 & 1 & 1 & 0 & & & & & = & v^{(3)} \\
\hline
1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & = & p
\end{array}
$$

n=5
n-1, n-1, n
4
4
5
4
5
6
6
6
7
7
7
8
n, n, n → (n, n+1)
(n-1, n-1, n) → (n, n)
(n-1, n, n+1) → (n+1, n+1)
m,

My n is 5 I know say I am adding n minus 1, n minus 1 and n. So, the first number is 4 bits, second number is 4 bits, third number is 5 bits this is a maximum I can give. Even for this maximum value the sum is going to be 1 1 1 1 1 and the carry is going to be 1 1 1 1 0 because these two are 0 the carry is going to be 0 respective of this.

So, this is just n bits and this will also be n bits where is n is 5. So, this is 5 bits, and this is also 5 bits. So, if I am adding n minus 1, n minus 1, n minus 1 n I get n comma n; suppose I do n minus 1, n and n plus 1, please note that I will get n plus 1 and n plus 1; if I am adding n minus 1, n and n plus 1, yeah I have done it here now that we can prove that I will just spend few minutes to prove; suppose I am adding.

(Refer Slide Time: 31:58)



Carry Save Addition based Multiplication

$$\begin{array}{cccccccccc}
 & & & & 0 & 0 & 0 & 0 & = & 0 \\
 & & & & 1 & 1 & 1 & 0 & = & m^{(0)} \\
 & & & 0 & 0 & 0 & 0 & & = & m^{(1)} \\
\hline
 & & & 0 & 1 & 1 & 1 & 0 & = & u^{(1)} \\
 & & & 0 & 0 & 0 & & & = & v^{(1)} \\
 & & 1 & 1 & 1 & 0 & & & = & m^{(2)} \\
\hline
 & & 1 & 1 & 0 & 1 & 1 & 0 & = & u^{(2)} \\
 & & 0 & 1 & 0 & & & & = & v^{(2)} \\
 & 1 & 1 & 1 & 0 & & & & = & m^{(3)} \\
\hline
 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & = & u^{(3)} \\
 & 1 & 1 & 0 & & & & & = & v^{(3)} \\
\hline
1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & = & p
\end{array}$$

So, let n be 5. So, I am adding a 4 bit number with a 5 bit number, and the 6 bit number right.

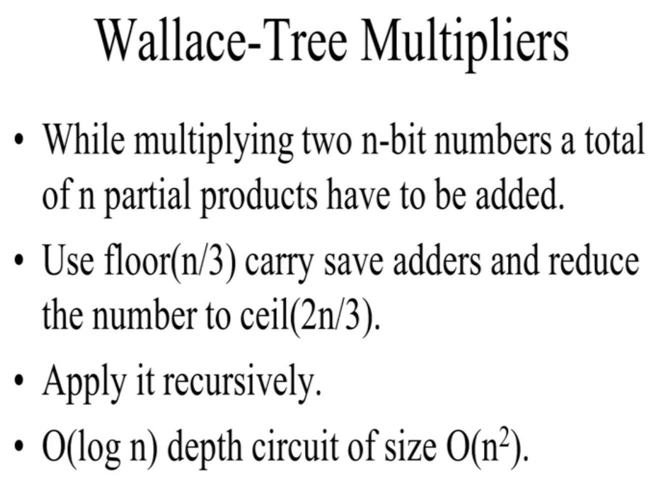
Student: (Refer Time: 32:10).

No, we will come this is there any doubt or query about this part. So, is there any doubt about this part no right, is there any doubt about what I am marked on red is there any doubt no. Now we will go to this the second n equal to 5, when n equal to 5 I am doing this second part I am adding n minus 1 that is a 4 bit number with a 5 bit number and a 6 bit number. Now what will be the sum bits here this will be 1 1 1 1 0 1, and the carry bits will be 1 1 1 1 1 0. So, this will be 6 bit number and this is also be a 6 bit number always this carry will be 0. So, when I add an n minus 1 and then n plus 1, I get two n plus 1 when I add just an n n n then I get n comma fine.

If I add an n and n I get an n comma n plus 1 bit number. So, kindly please remember this right. So, simple proof for it is in this case for example, let me try and prove this this is an n minus 1 bit number, this is an n bit number and this is a n plus 1, please note that since it is n minus 1 bit let us take the last 3 stages right n plus 1 n and n minus 1 the last 3 stages. So, here this number would be 0 and this will also be 0, the next this will be zero. So, then I will have some bits here. So, let me say these bits are a b c. So, this can upmost create one carry right. So, this can create one carry of one, and if this fellow is also one a is also one right irrespective of whether a is 1 or 0 no carry will be generated after this because the maximum I can have is right.

So, this can basically create me. So, that is what this happening here. So, whatever this creates will be a sum bit here, and whatever this creates as a carry is going to be one. So, since these two are zeros right, this this particular stage will never generate a carry; because two of the 3 bits are zeros. So, I cannot generate a carry. So, the carry will this always will be 0, I getting this right. So, that is a simple argument of how an n minus 1, n and n plus 1 can generate to n will generate only to n plus 1 bit numbers. So, these 3 equations are very important because we use this to actually reduce the size of the entire circuit. So, kindly remember this so that in the next class you know one of you please tell me. So, I will write it down.

(Refer Slide Time: 35:56)



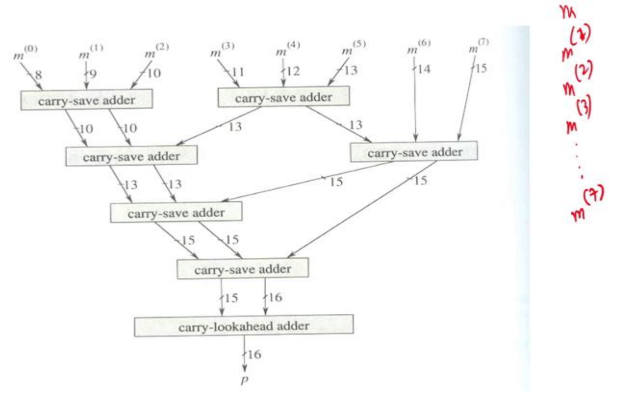
Wallace-Tree Multipliers

- While multiplying two n-bit numbers a total of n partial products have to be added.
- Use floor(n/3) carry save adders and reduce the number to ceil(2n/3).
- Apply it recursively.
- O(log n) depth circuit of size O(n^2).

This is the construction I will go and talk about this construction in great detail in the next class, but just to give you sum.

(Refer Slide Time: 36:04)

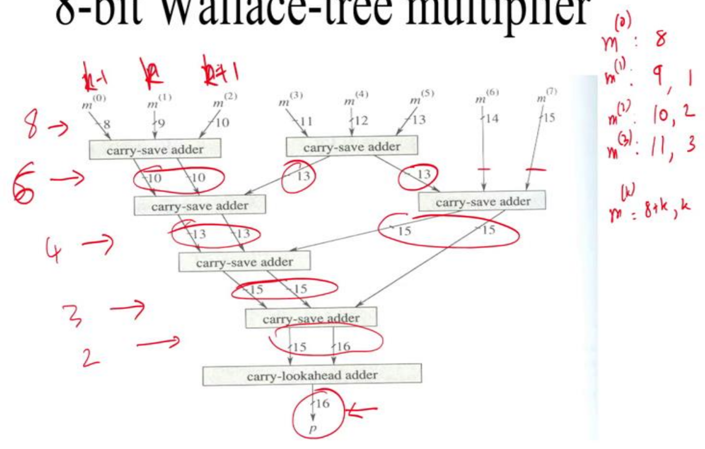
8-bit Wallace-tree multiplier



So, I am trying to multiply two 8 bit numbers right. So, I will basically get m 1, m 2 sorry m 0, m 1, m 2, m 3 till m 7 right. So, m 0 right these are all the partial products, I will get 8 partial products m 0, m 1, m 2 till m 7 that I will get it in constant time right using just I have to and each bit with the of the multiplicand with the multiplier or of the multiplier with the multiplicand.

(Refer Slide Time: 36:54)

8-bit Wallace-tree multiplier



So, what is m 0? M 0 is an 8 bit number, m 1 is a 9 bit number with 1 0; m 2 is a 10 bit number with two zeros at the last, m 3 is a 11 bit number with 3 zeros. So, m k is the 8 plus k bit number with k zeros at the right hand side right, most k bits are zeros right; now I have 8 numbers how can I add it?

So, I will take m 0 m 1 and m 2 right m 0 is an 8 bit number m 1 is a 9 bit number, m 2 is a 10 bit number. So, if I add 8 9 and 10 bits I will get two 10 bits, if I have n, n minus 1 and n plus 1 I will get two n plus 1 bits where n is 9. So, I get two 10 bits, because I am adding n, n minus 1, n and n plus 1, or k minus 1, k and k plus 1; if I add 3 numbers I will get two k plus 1 bit numbers so I get 10, 10. Now I am adding m 3 m 4 and m 5, m 3 is a 11 bit number, m 4 is a 12 bit number and m 5 is a 13 bit number, when I add again 11 12 and 13 I get two 13 bit numbers, I get two 13 bit numbers so.

So, what I have done is I had say totally 8 numbers to add here right I have now made it as what 5 numbers, I took the first 3 numbers and made it to. I took the second 3 numbers and made it in to 2, and then there are two more left. So, now, I am made it as what 1, 2, 3, 4, 5, 6 sorry 6. So, 8 numbers I actually made it into 6, and how much time I require to convert this 8 numbers into 6 numbers, how much time did I take? Constant time because carry save adder takes constant time right. Now this 6 numbers now I will take these 3 numbers 10, 10 and 13. So, 10 could be again 10 can be treated as this same k minus 1, k plus 1 right.

Student: (Refer Time: 39:26) 13 (Refer Time: 39:36) 11 and 13.

No 10 10 13 I can add considering it as each one is as a 10 bit number, another is as 11 I can put up and sum zeros there.

Student: Will be getting 11 and 13 numbers (Refer Time: 39:44) 1 11 bit and another (Refer Time: 39:44).

No for the time being will not complicated because I have to go and talk about optimization later. So, when I add 10 10 and 13 I will get two 13 bit numbers, the 10 bit number could be treated as a 11 bit number, this 10 bit as a 12 bit number, and this 13 as 13 bit number. So, I can still map this 10, 10, 13 to k minus 1, k and k plus 1. So, I will get two 13 bit numbers, now I have a 13, 14 and 15 again k minus 1, k and k plus 1. So, I will get 2 15 bit numbers. So, this 6 actually became 4 in constant time; now this 13, 13 and 15 will give me 2 15 bit numbers and I have another 15 bit numbers.

So, these 3 15 bit numbers I can add to get. So, this 13 13 15 I can add I will get two. So, this 4 became 3 here and this 3 with this 3 actually became two here, and If I have two numbers I can add it get this value. So, lot of redundant hardware is inside this circuit, but one of the thing I need to prove to you is that when I multiply two 8 bit numbers I will get a 16 bit number as an answer right. If I did not do if I just say that it is 8, 9 and 10, I say this is 10 bit and 11 bit right in our thing right if I add 3 n bit numbers then I get a n and n plus 1 bits. So, if I just go and say this 8 9 10 I can consider them as 10 bit numbers right and if I add 3 10 bit numbers then I can say that this will get 10 and 11 bits while this will get 13 and 14 bits and we start expanding this answer will not be 16 bits.

It will be much larger right we will see zeros only, but when I just delineate this when I just expand this it will become an 19 bit or a 20 bit; it will not become 19 or 20 bit that is why I spent some time to explain you the previous slide where this if everything is not n then I will get only n n n or n plus 1, n plus 1. So, that is this is what we basically saw in the previous thing. So, from that if you see that 8 9 10 will give me 10 10 11 12 13 will give me 13 13 14 15 is there, now this 10 10 13 will give me 13 13 this will give me 15 15 and then I add this and I get this ok.

So, whenever I get a 16 bit number as an answer right. Now what we will do in the class is two things we need to do what is the time complexity. So, now, you will see this is a tree right. So, almost a tree that is why we call it as a Wallace tree multiplier, now we see this is a tree. Now what is the height of this tree? The height of this tree is the depth of the circuit right. So, we will now go and prove what is the height of this tree and also we will put some effort to see how I can go and reduce the redundancy here. So, lot of zeros get added repeatedly; in this 11 12 and 13 there are 3 three zeros that get added here. So, now, how do we adjust those zeros?

So, how can I construct a much more optimize circuit then this right we will see that this, this I have put here to explain you the concept most importantly to tell that when I add to a multiply to 8 bit number I will get a 16 bit, if I multiply to n bit numbers I will get the 2 n bit or if I multiply m bit number with a n bit number I will get m plus n bits as an answer fair enough. So, we will continue this in the next class.