

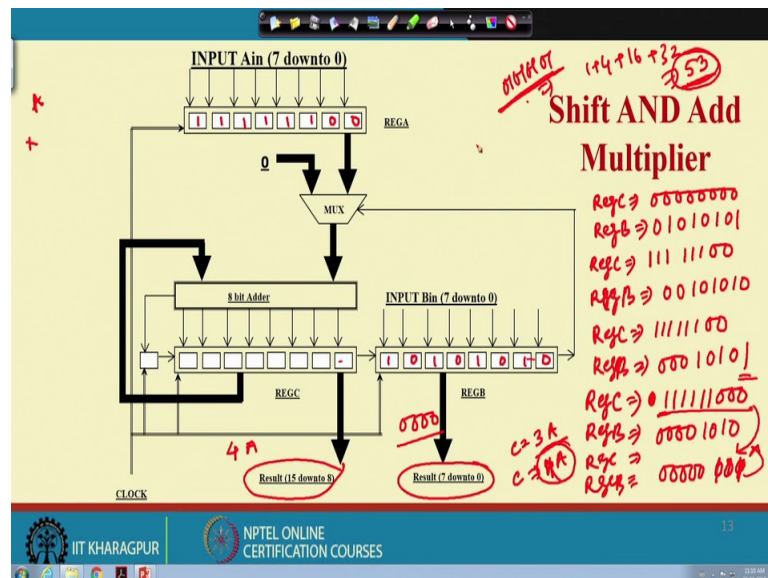
**Architectural Design of Digital Integrated Circuits**  
**Prof. Indranil Hatai**  
**School of VLSI Technology**  
**Indian Institute of Engineering Science and Technology, Shibpur, Howrah**

**Lecture - 28**  
**Multiplier Architecture ( Contd. )**

Welcome back to the course on Architectural Design of IC's. So, we are discussing in the last class about the Multiplier Architecture. So, we have seen this, the basic multiplication operation which is the serial multiplier architecture. So, apart from that in combined with parallel multiplication operation; that means, parallel as well as this serial, how we can do that type of architecture also we have seen ok.

So, then again what I said that there is the basic multiplication of operation is nothing but this shift and add ok. So, that has the basic fundamentals of any multiplication operation. So, that type of multiplier architecture now we will see; so, where if this is nothing but that shift and add ok.

(Refer Slide Time: 00:57)

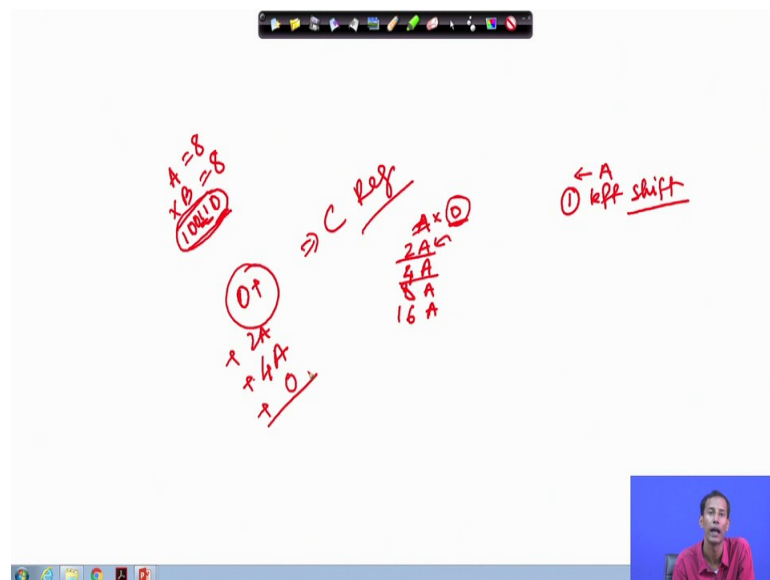


So, here you see I am having 2 inputs which is A in and B in ok. So, each of if I consider each of them are like a 8 bit ok. So, I have I am having 8 bit register over here for input A in and I am having 8 bit register for B in.

So, based on that I am having one adder over here one adder over here ok. So, where this adder will again one of the input to the adder that is the corresponding shifted bit of this B and this another; that means, sorry not shifted bit of this. So, this shifted that means, this addition operation they are basically controlling the, this B in. How? What is the how basically this circuit is operating that I am just coming later. So, I am having 2 registers of A in and B in and I am having one adder ok. So what I need? Shift and then add ok.

So, then addition operation this after this getting the sum they will be stored somewhere that is this particular registers. Again what I basically need to add? What I need to add? Ok.

(Refer Slide Time: 02:47)



If I consider this particular case, so, if I need to; that means, do this addition operation, suppose A into B I have to do; A of 8 bit and B of 8 bit right. So, at that time what will happen? Initially what we will do? I have to I will get A, ok. Considering suppose if I consider all the bits of B are 1. So, at the time, how we can do? At the in next I will get 2 A then I have to add 4 A then I will get 8 A then I will get 16 A, something like this I will get. That means what?

I am shifting the A in for 1 bit left shift and if this bit is 0, if any of this bit is 0, so at that time it will consider not A it will consider or it will choose 0. So that means what I am doing? Suppose for this particular case, so, initially it will not be A, it will be 0 and in

the next if it is 1 so at the time I have to get 2 A and that will be come to the adder; that means, this 0 will be come at first then that will be added with 2 A.

Then in the next if this is 1, so at the time this 0 plus 2 A, is already saved in C register in the previous particular; that means, architecture. Then if I consider this 1 so, at that time what will happen? This will be shifted by 2 bit, 2 bit left shift; that means, at the time I have add 4 A. If this bit is suppose this bit is, suppose this bit is 0, so, at that time this will be added with is 0. So, that is why if I just go back this particular case, so you see whenever this basically what is happening? This is the input B in, B in ok. So, B in is loaded over here and A in loaded over here right.

So, initially, if I add for this previous example what I consider that let us consider the first bit is 0, ok so that means, this bit is 0. So, that will come over here and it will choose 0 instead of A. So, then 0 is the 1 input and it is connected to the feedback; so that means, now 0 plus 0, so this will 0 will be stored over here ok. So, then again that bit will be; that means, if the next bit is 1, suppose this is initially it was 0 then it is 1 ok. So, then 1 coming over here so, it is choosing this particular A in this particular A in and then 0 that is again added with this particular case and here whenever we are choosing this A in over here, not that means, based on this what value of A. So, what is the weight of this that I need to calculate whether that will be 2 A or that it will be 4 A?

So, here using this MUX, I can compute that ok, so for 0 this will choose 0. If this bit is 1, so at that time it will choose 2 of A; that means, whether the A in will be just shifted by 1 left shift and that will come over here and that will be stored in this particular register ok. So, in this manner now in this manner that will be set and, how we are doing this particular? Suppose for this case if I consider suppose, if I consider suppose for A that is 11111100 and let us consider 10101010 ok. So, initially what is happening? 0 is coming. So, this is filled with all 0. So that means, this register C at the very beginning it is filled with all 0 ok.

So, then it is choosing A; that means, this 1 ok. So, that now or this is 0. So, this 0 is basically coming over here ok. So, then this bit is shifted; that means, at that time this REG A sorry REG B will be 01010101 ok, then 1 is basically coming over here and it is choosing this A. So, here initially it was 0.

So, at the time this REG C is nothing but your 11111100 ok. So, then again this 0 is basically coming over here. So, register B will be 00101010, then again it is 0; means it will choose 0. So, at that time 0 added with this. So that means, now REG C will not change its value correct, so that means, it will remain 6. Then again this 0 will come over here; so that means, now register B, the value of register B that will be 00010101, then again this 1 will come out and again it will choose A, at this particular case. So, at that time what will be the values of register C? That will be A plus A; means what? That is twice of A ok. So, A plus A means twice of A; so that means, now it will be sorry, it will be 111111000 ok, so; that means, now this 1 will this 1 will come over here ok.

So, now, again this 0 will come; so; that means, now at that time the register B that will be 0000 then 1010. So, then again 0 means register C will remain same. So, it will be come over here. So, register B again that will be changing its value to five 0 then 010 sorry 101 ok. So, then again A one of this will come ok, so that means, again I have to add this with A again. So that means, now C register will be thrice of A and this will be in this manner it will be this C will be finally, that will be of 4 A, here it will be 4 A and all these bits will be 0 ok.

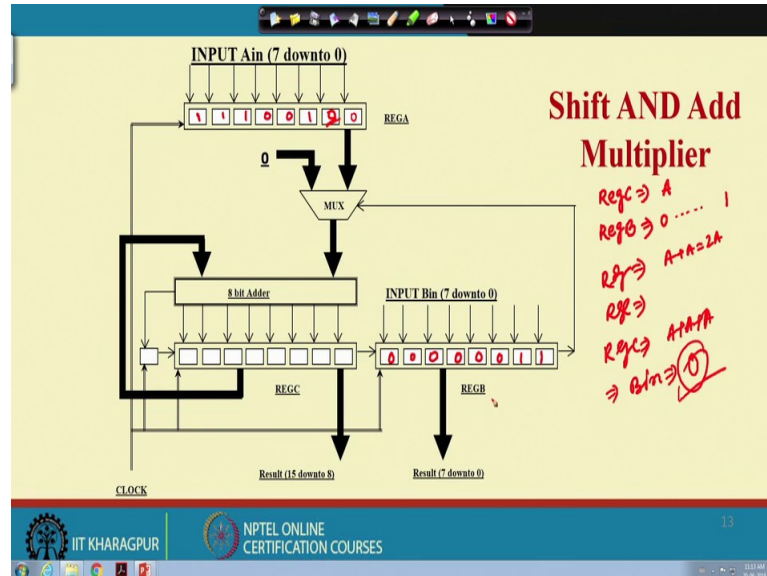
So, now if you if I; that means, the results whatever is the remainder over here ok. So, that is the results of 15 down to 8 and the results over here which is remaining over here. So, that is the results 7 down to 0. That means what? Now, if I just do it in other way; that means, whenever I am multiplying; suppose this is A, I am multiplying with whatever what is the value of this? That is 1 then 2 sorry 1 then what is the value of this? 01010101. So, the value of this is 1 plus 4 plus 16 plus 32; so that means, total 53 ok.

So, in this manner now if I just; that means, calculate ok. So, actually I have to draw it will not come as 4. So, finally, what value I will get over here ok, so that will be shift in as I am doing the shifting operation in this particular as in this case; so, it will give me the corresponding weight of 53 as I am accumulating A over here in this case. So that means, whenever I will shift this based on this will be shifted and then again it will come over here ok. So, in this manner I can do this shift and add operation.

Actually, if I consider one another example at the time it will be much more clear. So, whenever we are following, so at that time you choose any number over here and you

chose any number over here and you do; that means, how this operation basically is happening and then you see whether you are getting the correct results of that or not ok.

(Refer Slide Time: 13:57)



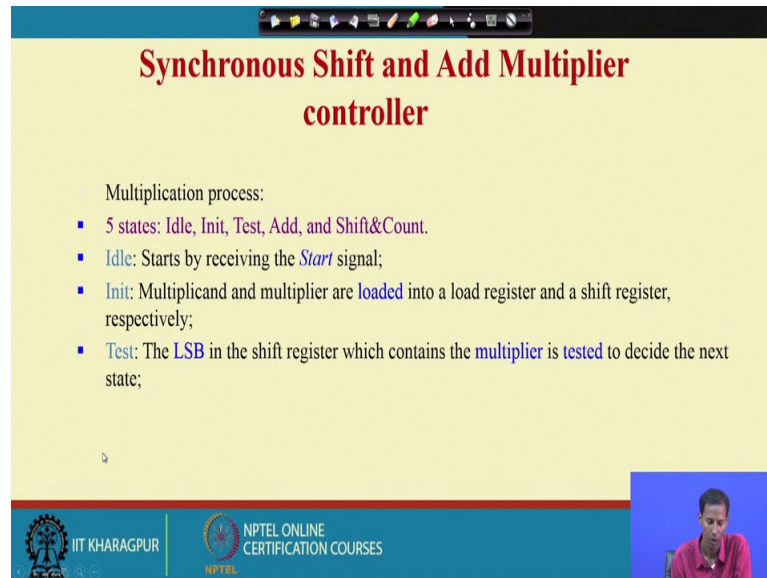
So, suppose let us consider another example. If I choose only this value 3 values if I consider for a smaller value I think at that time I do not need to that much of. So, if I consider 3 over here. So, at a time what will be the register C? So, the register C will be it is 1 and it is it will chose A; so that means, it will be A and A means what? Whatever is the values of value of these that will now come to this particular position right ok.

So, then what will be the register B? If I choose that as suppose this value of this is so, the register value of; that means, this will be all 0 and that will again shifted to this particular case ok. So that means, now again this will be added. So, initially 1 it is choosing this A ok, then again this is 1 then again that is register A will be A plus A that will be of 2 A.

Then again in the next this, what will be the register B value? If that is 0, then that 0 will come over here and then again it will try to push this 1 and then again it will come down to this, it will choose at the final this register C will be added with A plus A plus A and all the corresponding B in value that will be all 0's or that whatever is the values of this that A in will try to fill this position ok.

So, in this manner in this fashion we can; that means, do this shift and add multiplication ok. So, it will be better if you choose your own particular numbers over here and then you try to calculate the corresponding multiplication operation ok.

(Refer Slide Time: 16:40)



**Synchronous Shift and Add Multiplier controller**

Multiplication process:

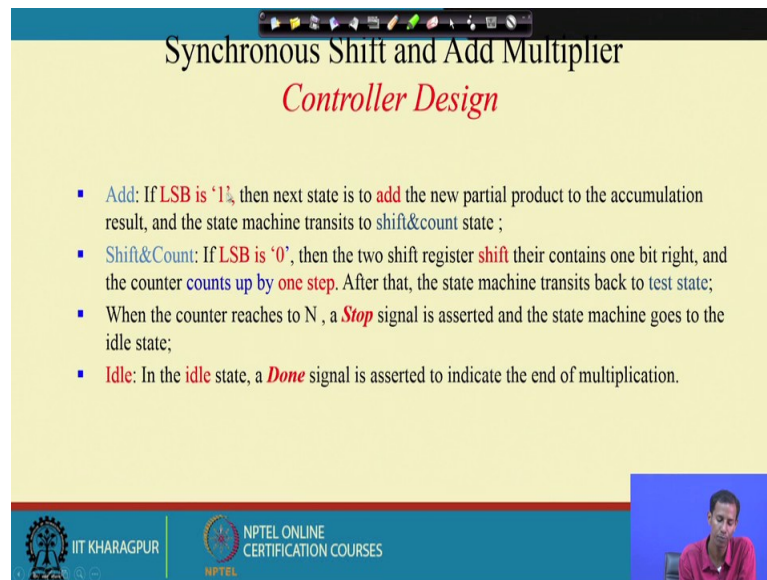
- 5 states: Idle, Init, Test, Add, and Shift&Count.
- Idle: Starts by receiving the *Start* signal;
- Init: Multiplicand and multiplier are loaded into a load register and a shift register, respectively;
- Test: The LSB in the shift register which contains the multiplier is tested to decide the next state;

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, then so, we have this synchronous shift and add multiplier. That means, whenever we are doing this multiplication shift and add, so, at that time there are 2 type of multiplier with one is synchronous another one is asynchronous ok. So, in synchronous add multiplier is very much used; that means, common in used. So, they have 5 states which are idle, initial, tests, add and shift and count.

So, in the idle state it starts by receiving the start signal; that means, whenever if this starts signal will be 1 or high, it will start of the operation of the corresponding multiplication. And in the initial case the multiplicand and the multipliers are loaded into the load register and a shift register respectively. That means what? In each case this A whenever I in initial signal is high, so, the A in and the B in that will be loaded into 2 corresponding register. Then this test signal, the LSB in the shift register which contains the multiplier is tested to decide the next state ok.

(Refer Slide Time: 18:20)



The slide is titled "Synchronous Shift and Add Multiplier Controller Design". It contains a list of four bullet points describing the state transitions of a state machine. The first bullet point describes the "Add" state where the LSB is '1', the new partial product is added to the accumulation result, and the state machine transitions to the "shift&count" state. The second bullet point describes the "Shift&Count" state where the LSB is '0', the two shift registers shift one bit right, and the counter counts up by one step, after which the state machine transitions back to the "test" state. The third bullet point states that when the counter reaches N, a "Stop" signal is asserted and the state machine goes to the "idle" state. The fourth bullet point states that in the "idle" state, a "Done" signal is asserted to indicate the end of multiplication. The slide also features logos for IIT Kharagpur and NPTEL Online Certification Courses, and a small video inset of a person in the bottom right corner.

Synchronous Shift and Add Multiplier  
*Controller Design*

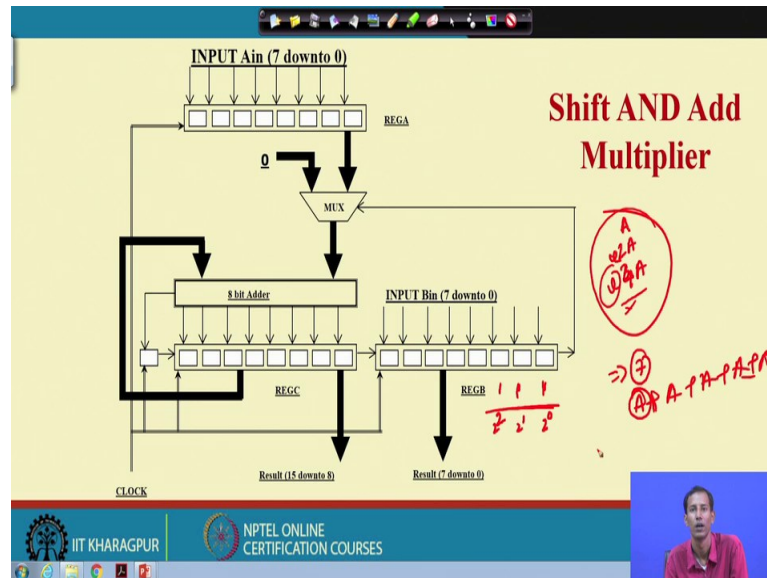
- **Add:** If **LSB is '1'**, then next state is to **add** the new partial product to the accumulation result, and the state machine transits to **shift&count** state ;
- **Shift&Count:** If **LSB is '0'**, then the two shift register **shift** their contains one bit right, and the counter **counts up by one step**. After that, the state machine transits back to **test** state;
- When the counter reaches to **N**, a **Stop** signal is asserted and the state machine goes to the **idle** state;
- **Idle:** In the **idle** state, a **Done** signal is asserted to indicate the end of multiplication.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, if LSB is 1, then next state is to add the new partial product to the accumulation results and the state machine transits to shift and count state ok. That means, what I said that whenever; that means, this LSB of this is 1. So, at the time what I have to do? I have to do this addition operation and shift and count; that means, this shift and count also I have to perform right.

Then in shift and count if LSB is 0, then two shift register shift they are contains one bit right and the counter counts up by one step. After that, the state machine transits back to test state. When the counter reaches to N, a Stop signal is asserted and the state machine goes to the idle state and in idle state in the idle state, a Done signal is asserted to A indicate the end of the multiplication operation ok. So that means, now what is happening? How many times I am the basically doing this 1 over here, I am getting 1 over here? So, that is being counted.

(Refer Slide Time: 19:59)

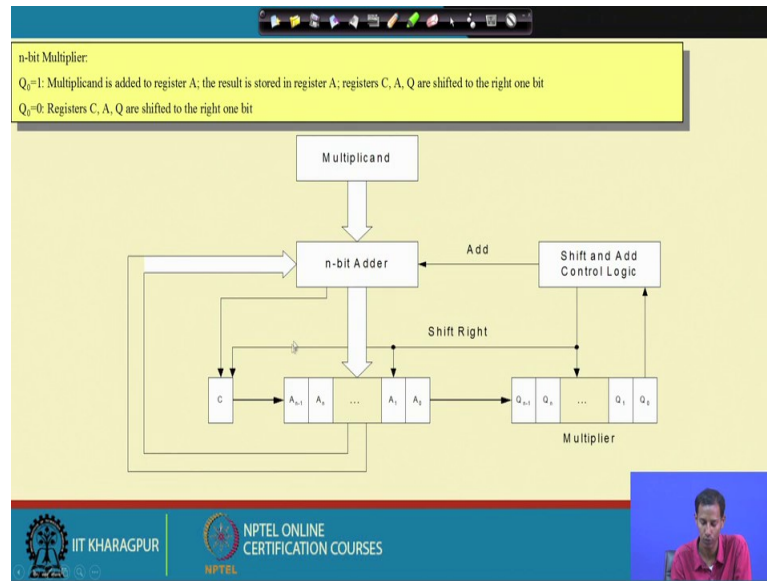


Why I need to count that? Because based on that what I said? That I have to add something like these A and then 2 A and 3 then 4 A something like this I have to add. So, how many times I am basically adding, so, that I have to do. So, this is what I am getting over here? In this particular case, how many times? So, if I am getting 3 times 1 over here, so that means based on this I will get the corresponding weight. So that means this is what? This is 2 to the power 0, then two to the power 1, this is 2 to the power 2. So, total 7 number of shift I have to do it in A or 7 times I have to add A something like this ok.

So, that is why based on this sorry based on this particular case now I can; so, this is the controller design; that means, if this is the condition if the LSB of B is 1 then add and then shift and count in shift and count if the LSB is 0 then do the that means, go to the that means, the counts up by one step. If it the counter reaches to N; that means, what is the, that N is the number of bit or if it is I in this particular case that we have consider that weight. So, if it is reaches to n, so, at that time stop this signal and whenever this stop the signal; that means, finally, the results is done you just break the loop or come out to the output ok. So, these is the shift and add multiplication operation.



(Refer Slide Time: 21:59)



So, here you see this is the multiplicand then there is a n-bit adder and then there is this shift and add control logic. So, this is the corresponding A and this is the corresponding this B, here it is considered as Q. So, whenever this Q 0 is basically it is checked in each of these things. Whenever this is 1, so at the time it will follow that logic what I discussed and if it is 0 then need to be followed different logic ok. So, based on that it will be just added something like this in a repetitive manner ok so, this is nothing but a synchronous shift and add multiplier, the same thing ok.

(Refer Slide Time: 22:52)

### Comparison between Synchronous and Asynchronous Approaches

Multiplicand	Multiplier	Total calculation time (ns)	
		Synchronous	Asynchronous
0110	0000	66	43
0110	0010	72	45
0110	1010	78	46
0110	0111	84	48
0110	1111	90	50
Average power consumption (mW)		5.1	2.7
Total chip area (mm <sup>2</sup> )		0.0855	0.1075

So, and then what I said that I am having 2 type of multiplication, what is synchronous another 1 is asynchronous ok. So the, this is the basically this computation time requirement in synchronous as this is happening that means, all the things are happening clockwise. So, that is why I have to wait for more time in synchronous, otherwise actually I have not discussed the asynchronous circuit here, but if you follow the asynchronous circuit for this; that means, some of the changes or some of the that shifting operation that you have to; here what is happening?.

Basically this in particular clock, it is finding the LSB position of B, then not finding it is checking the position of B and then it is tries to do this shifting operation and count operation; that means, added by this 1 and again it is also doing that addition operation 2 on a same clock edge. But in asynchronous design, you can do it in a different way. So, that means that will not happen in one particular clock cycle ok. So, that is why this time requirement in asynchronous case that is lower than the synchronous case ok. But then what is the problem in asynchronous case?

So, you can get; that means, the whenever you are doing operation, so at that time you can get more violation; that means, more violation means, if one of the signal is not coming at to any particular gate at proper time, so at the time there will be mismatch in the results. So that means, if the data is not properly aligned, so, at that time you will get erroneous result or; that means, garbage value at the output, which will not happen in the case of synchronous circuit, but which can happen or which there is a high probability to for happening the that type of things in asynchronous circuit.

But what I am getting? I am getting the competition time is much more lower than the synchronous circuit ok. And apart from that the circuit chip area that is also more in this case of asynchronous circuit as we are putting some extra logic to precompute the values ok. So, this is the comparison between this synchronous and asynchronous circuit. Then, so these up to this is this shift and add based multiplication. So, what we have learnt? Serial multiplication, then serial plus parallel multiplication, then shift and add based multiplication. So, in shift and add based multiplication there are synchronous multiplication and asynchronous multiplication, then there are this array multiplication.

(Refer Slide Time: 26:04)

**Array Multiplier**

- Regular structure based on add and shift algorithm.
- Addition is mainly done by carry save algorithm.
- Sign bit extension results in a higher capacitive load and slows down the speed of the circuit.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

19

So, array multiplication is also the regular structure which is based on this shift and add based algorithm ok. So, here the main thing is that this addition is mainly done by carry save algorithm or carry save addition operation and then sign bit extension results in a higher capacitive load and slows down the speed of the circuit. So that means, why I need to if I doing if I am doing this sign multiplication, so at that time I have to consider this sign bit extension; otherwise, I do not have to ok. So, whenever I am doing this using this array multiplication I can do sign multiplication as well as unsigned multiplication, but whenever I am doing a unsigned multiplication at that time I do not need anything. But whenever I am doing sign multiplication, so at that time I have to be bother about signed extension.

So, how we do signed extension that we will come later ok. But whenever we are doing this sign bit extension in array multiplication, so at that time; that means, I am putting extra overhead to the circuit ok, which basically degrades the performance of the or there that is the disadvantage of this particular architecture ok. So, we will discuss more on to this on the next class.

Thank you for today.