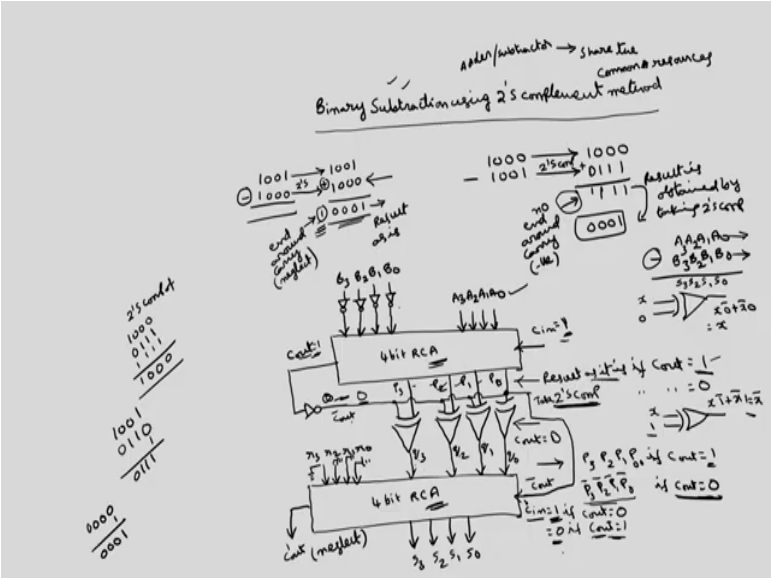
System Design Through VERILOG
Prof. Shaik Rafi Ahmed
Department of Electrical and Electronics Engineering
Indian Institute of Technology, Guwahati

Gate level modeling - I
Lecture - 05
Parallel adder/subtractor

So, in the last class we are discussing about the binary subtraction using 1's complement and 2's complement methods. So, in that most of the processor uses signed 2's complement method because if any intermediate overflows are present, then we can simply neglect those. So, today we will discuss about the architecture or circuit diagram of this binary subtraction using 2's complement method.

(Refer Slide Time: 01:01)



So, as we have discussed, in binary subtraction using 2's complement method. If we want to perform say, 1 0 0 1 minus 1 0 0 0. So, we have taken minuend as it is 1 0 0 1 subtract and we have take the 2's complement of this.

So, the 2's complement of 1 0 0 0 is 1's complement is 0 1 1 1 and 2's complement we will have to add 1. So, 1 0 0 0 is same. 2's complement and the original number both are same in this particular case.

So, now the subtraction will convert into addition we perform this addition bit by bit 1 0 0 this is 1 0, this is called end around carry also called as overflow. So, if end around carry occurs then we have to regulate it and result is positive.

So, this itself is result this is result as is. So, on the other hand if we have 1 0 0 0 minus 1 0 0 1, we have to take the subtrahend minuend as it is. And subtrahend, we have to take the 2's complement. So, the 2's complement of 1 0 0 1 will be 0 1 1 0 is 1's complement we have to add 1 so, 0 1 1 1.

So, you have to add 1. So, you will get 1 1 1 1 no end around carry. So, it says that the result is negative. Result is obtained by taking the 2's complement. So, what is 2's complement of 1 1 1 1, 0 0 0 0 is the 1's complement, 2's complement becomes 0 0 0 1. So, this is 2's complement as 0 0 0 1 this is the result.

Now, how to implement this algorithm using digital circuit or what is the architecture corresponding to these examples. So, in one case, in subtraction we have to take the complement of 2's complement of subtrahend ok. So, for that, so we need a 4-Bit Ripple-Carry Adder that we have discussed in the last class.

This is 4-Bit Ripple-Carry Adder. This we have two sets of the numbers. If we assume that these numbers are in general, I want to perform A 3 A 2 A 1 A 0 minus B 3 B 2 B 1 B 0. Then, I have to take to mean I implement this using addition I will take this as it is I will take the 1's 2's complement of this one I will perform the addition ok.

So, I will take A 3 A 2 A 1 A 0 as it is. Then, I will take B 3 B 2 B 1 B 0 through NOT gates. So, this will give 1's complement, if I pass through the NOT gates. This is B 3 B 2 B 1 B 0, but what is required is 2's complement.

So, this has to be added, these two set of the numbers has to be added. So, you make this C in as 1, so that we will get 2's complement. If this is 0, 1's complement. If this is 1, 2's complement. Then, I have got the minuend as it is and subtrahend we have taken the 2's complement of subtrahend by adding 1.

Now, after adding this, so we will get 2 conditions. This you can call as C out this is end around carry. So, end around carry can be either 0 or 1. So, if end around carry is 0, you have

to take the 1's complement of the sum. So, we will get sum here ok. We have taken the 2's complement of the sum. If end around carry is 1, then simply we have to neglect the end around carry result is as it is whatever the result that we have obtained here is the correct result.

This result we have to take as it is, if C out is equal to 1. And we have to take the 2's complement if C out is 0. So, before going for this implementation of this logic, I will just give a simple circuit exclusive OR gate ok.

So, if I take exclusive OR gate if 1 input is x other input is 0, what will be the output? So, this is x into 0 bar plus x bar into 0. So, this is nothing, but x. So, on the other hand, if I take one input as x another input as 1, then what will be the output? x into 1 bar plus x bar into one. So, this is nothing, but x bar that is for the exclusive OR gate if one input is 0, the output will be same as the other input.

And similarly, if one input is 1, the output is complement of the second input ok. So, we have to take for exclusive OR gates, because in one case I want the result as it is in the other case, I want the complement of the result. So, that can be implemented by using this exclusive OR gates.

So, one input will be the sum output that will be obtained from this 4-Bit Ripple-Carry Adder ok. So, what about the other signal? This is the final result. If I call this final result as S 3 S 2 S 1 S 0 subtraction result. So, this is S 3 S 2 S 1 S0, but what will be the second signal?

So, the C out we have to use. So, if C out is 1, we have to take result as it is, but if one input is there normally, for exclusive OR gate will get complement so, you have to pass through this C out through the NOT gate.

So, that whenever C out is equal to 1 this is 0. So, 0 means as it is. So, whenever C is equal to 1 result is as is ok. So, this we have to give as a common signal for all the exclusive OR gates as a second input. So, this will perform the 4-Bit binary subtraction. So, we have take the 2's complement. So, with this logic you are getting the 1's complement of this this is not the final result. So, if C out is equal to 1, then this will be 0 you will get as it is values. If C out is 0, we will get the complement of these values.

If I assume that this is some p 3 p 2 p 1 p 0. So, the output here will be p 3 p 2 p 1 p 0 if C out is equal to 1, because if C out is equal to 1, this is 0. So, the other input is 0 means the same p 3 p 2 p 1 p 0 will be the output.

And if C out is 0, this will be p 3 bar p 2 bar p 1 bar p 0 bar, but how to get the result? We have taken the 2's complement. This is only 1's complement only. So, we need to connect 1 more 4-Bit Ripple-Carry Adder here. So, one set of the inputs are this, if I call this one as q 3 q 2 q 1 q 0.

So, this q 3 q 2 q 1 q 0 can be either p 3 p 2 p 1 p 0 are complement of that depends upon this C out. Now, we have to use another 4-Bit Ripple-Carry Adder. This is just to add 1 or to add 0 ok. So, this if I call as C in of this 1 this is C in of the first order if we call as this as C in `.

So, to get the 2's complement, whenever this p 3 p 2 p 1 p0 comes as a bar that is when C out is equal to 0. When C out is equal to 0, this should be 1. If C out is equal to 0, because C out is equal to 0 we need the 2's complement 1's complement we have obtained by using these exclusive OR gates and to get the 2's complement I have to add 1.

Of course, I will get the second set of these signals as permanently 0. This I will apply as q 3 is over r 3 r 2 r 1 r 0 if I assume. So, this actually I will connect all to 0. So, this I will ground all these signals. So, that if C in` is equal to 1 1 will be added to q 3 q 2 q 1 q 0 if C in` is 0, 0 will be added to q 3 q 2 q 1 q 0.

So, what will be the C in ` now? So, if C out is equal to 0 we have to add 1 if C out is equal to 1, I have to add 0 ok that same logic you can obtain here. So, if C out is equal to 1 here. So, this output will be 0. So, C out is equal to 1 means this is 0. So, that is what I want if C out is equal to 1. So, I have to add 0.

So, this directly I can connect this C out signal to this C in this is nothing, but C out bar. So, whatever the C out bar that we are going to obtain here we have to connect here. So, that when C out is equal to 0, C out bar is 1. So, 1 will be added and C out is 1 C out bar becomes 0. So, 0 will be added.

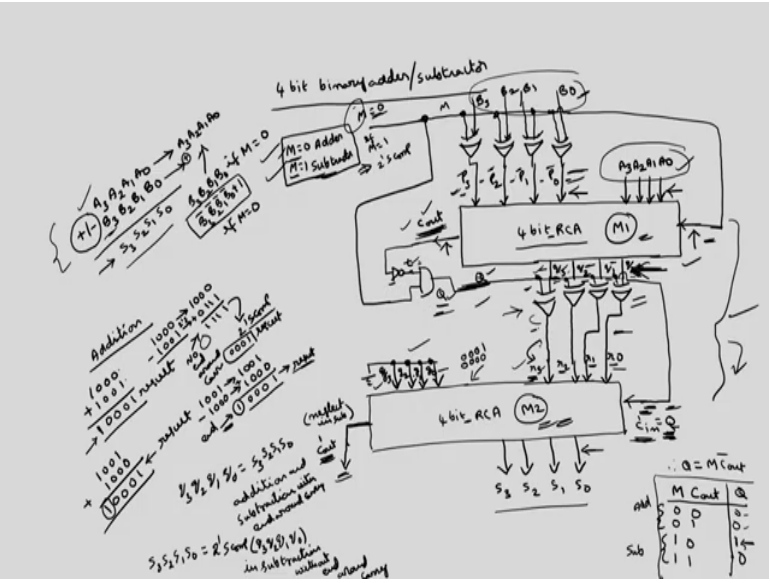
So, C in ` is simply C out bar ok. Then here, we will get the final sum outputs. This is S 3 S 2 S 1 S 0. We will get sum C out` say this we are going to neglect ok. This is how you can perform the binary subtraction using 2's complement method ok.

So, I will not write the VERILOG code, because in most of these processors, we will not have this binary subtraction circuitry separately. Instead, we have only one circuit which is capable of performing both addition as well as subtraction. That circuit we will call as adder/subtractor, because most of the models are common.

So, instead of using individual models for the adder and subtractor. So, you can play some logic where we can combine these two into a single block by sharing the common resources; that is the idea behind this adder subtraction. So, we are going to share the common resources.

So, I will discuss about this adder subtractor then, I will write a Verilog code correspond to the adder subtractor by instantiating this 4-Bit Ripple-Carry Adders, because in the earlier lecture we have discussed how to construct how to write the Verilog code for correspond to 4-Bit RCA ok using full adders then full adder using half adder and so on ok.

(Refer Slide Time: 15:21)



Now, what is the modification that you have to do here? So, now, we consider 4-Bit binary adder/subtractor. So, we have similar sort of the circuit and I draw here the similar sort of the

circuit. So, we require 2 4-Bit Ripple-Carry Adders. This is one 4-Bit Ripple-Carry Adder. I will call this one as module M 1 and I require one more 4-Bit Ripple-Carry Adder; M 2.

So, each Ripple-Carry Adder will be having 2 sets of the inputs. This is one set of input. This is another set of input and then, we have one carry in one carry out then, 4 sum bits. This is the structure of a 4-Bit Ripple-Carry Adder. Here, also, will be having 2 sets of the inputs. Then, you will get 4 outputs then, this is C out.

Now, how to inter connect these two 4-Bit R C A's? Again, we will come back to this example. Like if it is addition if this is 1 0 0 0 plus 1 0 0 1 then, simply how to add bit by bit 1 0 0 and this is 1 0. So, this along with this 1, this is the final result along with the carry. Even if it is 1 0 0 1 plus 1 0 0 0 also, same answer 1 0 0 01, this is the final result.

Whereas, in case of subtraction as we have already discussed. So, this 1 0 0 0 minus if we want to perform 1 0 0 1. So, we have to take 1 0 0 0 as it is and we have to take the 2's complement of this one and we have to add. So, this 2's complement of this one is 0 1 1 1. So, we will get 1 1 1 1. So, no end around carry so, we have take the again 2's complement of this to get the final result which is 0 0 0 1 is the result.

Whereas, here, if you want to perform 1 0 0 1 minus 1 0 0 0 this we have take as it is this we have take the 2's complement also, same thing. We have to add in 1 0 0 0 1, 1 end around carry. So, this as it is, is the result how to implement this logic is. Now, I have to use an extra control signal M is another control signal such that, M is equal to 0 this is the M is the control signal.

So, I am going to use a M control signal such that; M is equal to 0 this circuit will acts as the adder if M is equal to 1 the same circuit will acts as a subtractor. You can design in the other way also it is up to you. M is equal to 0 for the subtraction. M is equal to 1 for the addition again, design in that way also.

See here, I am taking this M signal here. Now, I am taking in general, this is I have to perform A 3 A 2 A 1 A 0 plus or minus B 3 B 2 B 1 B 0 and finally, I want this result as S 3 S 2 S 1 S 0. So, this is this can be either addition of these two numbers or subtraction of these two numbers ok.

So, in any case, if it is addition or subtraction this I have to take as it is A 3 A 2 A 1 A 0 as it is you have to apply to the 4-Bit Ripple-Carry Adder. So, I will take this input as A 3 A 2 A 1 A 0. Whereas, this second number you have to anyhow perform the addition whether, we have to take the direct B 3 B 2 B 1 B 0 or the 2's complement of this, that depends upon the M.

If M is equal to 0, I have to take the direct numbers. If M is equal to 1, I have to take the 2's complement of this number. So, here, this number will be direct B 3 B 2 B 1 B 0 if addition is nothing, but M is equal to 0 and this should be B 3 bar B 2 bar B 1 bar B 0 bar plus 1, that is 2's complement, if M is equal to 1.

So, how to derive the logic correspond to this. As I already discussed in the earlier binary subtraction exclusive OR gate will perform this type of operations ok. So, I will take here, B 3 B 2 B 1 B 0 will take exclusive OR gates. So, I will give one signal as this M signal.

So, if M is equal to 0, I have to keep as it is. I know that they are exclusive OR gate, if one input is 0, the other input will be the output ok this I am going to apply here is B 3 B 2 B 1 B 0.

So, if M is equal to 0, the same B 3 B 2 B 1 B 0 will be available here. If M is equal to 1, B 3 bar B 2 bar B 1 bar B 0 bar will be available that is 1's complement, but we want the 2's complement. So, what we will do is M is equal to 1, if M is equal to 1, I need the 2's complement. So, the same M we can connect to here clear.

So, M is equal to 1. So, here it will use the complement of these values here, at the same time it will add 1 also. So, that you will get the 2's complement of this B 3 B 2 B 1 B 0 will be added to A 3 A 2 A 1 A 0. Till now, we have obtained either sum of these two or sum of these plus 2's complement of this. Now, the actual logic appears here, depends upon the C out.

So, there are four possible conditions. So, ok this intermediate result is a final result in some cases and this intermediate result its a 2's complement will be the final result in other case ok, if I call this intermediate result as q 3 q 2 q 1 q 0 and final result is S 3 S 2 S 1 S 0, this q 3 q 2 q 1 q 0 should be equal to S 3 S 2 S 1 S0.

In which cases if this is addition and if it addition also we have to equal in subtraction also, if there is end around carry subtraction with end around carry, this will be equal to S 3 S 2 1 S 0 is equal to 2's complement of q 3 q 2 q 1 q 0 in only one case where, in subtraction without an end around carry how to implement this logic ok?

In 3 cases, this output is the final output in only one case, the 2's complement of this one will be the final output ok. So, for that I am using a simple logic here ok. So, M is going to decide whether it is an addition or subtraction and this C out is going to decide end around carry is present or not.

So, anyhow we have to use the exclusive OR gates here, because I have taken the 2's complement in one case. So, I give this exclusive OR gates. This is third exclusive OR gate, this is fourth exclusive OR gate. So, in 3 cases, as I have told the same input has to be transferred to the output. In only one case, the complements have to be taken at the same time this also, have to make as 1. So, that it will be 2's complement ok.

So, if I assume that the signal that we are going to connect the common signal. If I assume this common signal as say Q ok. If M is 0, what does it means? Addition in addition, regardless of whether you get these carry or not, I am going to take this q 3 q 2 q 1 q 0 as the outputs no need to take the complement.

So, regardless of the carry, carry can be 0 or carry can be 1 this is for addition. So, as it is you have to take means Q should be 0 if 1 input to this exclusion OR gate is 0 then as it is this q 3 q 2 q 1 q 0 will appear at the output of the exclusive OR gates. So, in addition regardless of the C out, I have to take the direct output. If this is one, so, this represents subtraction.

So, in subtraction also in one case, I have to take as it is. In the other case, you have taking the 2's complement. So, what is that case if carry out is. So, you see here, if there is no carry out, you have to take the 2's complement.

So, if carry out is 0, you have to take the 2's complement. 2's complement will be obtained by taking Q is equal to 1 has one input for the exclusive OR gate so that 1's complement will be obtained at the same time we have to add 1 also.

So in fact, this Q has to be applied to both. This 1 input of the four exclusive OR gates as well as the C in` or C in of the second carry look ahead adder ok. This should be same as Q, because whenever you want to take the complement of this q 3 q 2 q 1 q 0 that is 1's complement at the same time we have to add 1 so that we will get 2's complement.

If this is 1, means there is end around carry. So, result will be as it is. So, this Q is equal to 0. So, what is the Boolean expression for Q now, using this logic? You can easily write down the Boolean expression for Q. Therefore, Q is equal to only one min term is there. So, this is M C out bar.

So, simply Q can be obtained using this M signal, this M signal is one input and C out you have take through NOT gate, you obtain through the AND gate. This output has to be given as a Q this is Q signal.

So, this Q signal will be the input of the 4 exclusive OR gate one input of the 4 exclusive OR gate and at the same time the C in ok. So, this logic will play. So, what about this second set of the inputs?

This should be all grounded. So that the circuit will add only either this will add 0 0 0 1 or 0 0 0 0 means, as it is this data will be transferred ok.

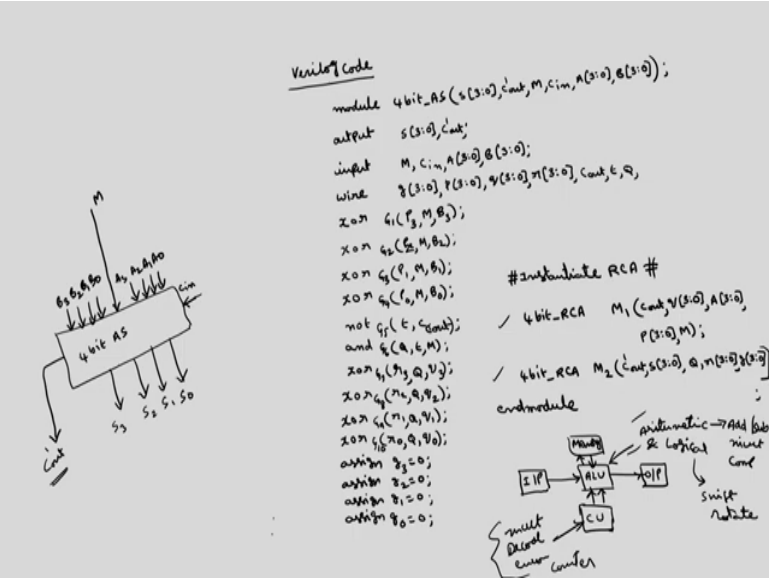
And this C out will be useful this final C out. We can call as C out` simply this will be this this C out`, in case of addition this there is a 5th bit. So, this along with this bit and these 4 bits is the final result in case of addition ok. Otherwise, in case of subtraction just simply you have to neglect this has to be neglected in subtraction and in addition this is the 5th bit of the result ok. So, this is the logic circuitry ok.

So, these are the final S 3 S 2 S 1 S 0. So, this is a 4-Bit binary adder/ subtractor ok. So, in the earlier lecture, we have discussed about how to construct a full adder using half adder. Initially, we have written Verilog code for the half adder, then we have discussed how to instantiate 2 half adders along with one OR gate to construct the full adder and using four such full adders. I have constructed the 4-Bit Ripple-Carry Adder.

Now, we can even instantiate this 4-Bit Ripple-Carry Adder the name that are given is 4-Bit_Ripple Carry Adder. So, 2 such models you can instantiate with some additional

circuitry to implement A binary 4-Bit binary adder subtractor now what will be the Verilog code correspond to this one. So, simply this circuit we are going to map on to the code.

(Refer Slide Time: 32:01)



So, if I take this block as 4-Bit adder/subtractor. This is 4-Bit AS adders subtractor ok. So, what will be the total final signals for this one? So, we have two sets of the signals A 3 A 2 A 1 A 0. And then, 1 final 1 input carry then, second set of the inputs B 3 B 2 B 1 B 0.

Then, we will get one final C out. We are calling this as C out` and then finally, we will get S 3 S 2 S 1 S 0 ok. This is equivalent to whatever the circuit that we have derived.

So, inside that block we have this complete circuitry. If I assume this as a complete block these are the 4 set of one input. This is 4 set of another input and M is also a control signal is there.

I will include that M also M is 1 additional signal and the remaining all are this C out this q these things are all internal ok this is also internal. So, this is final external C out` and S 3 S 2 S 1 S 0 remaining C in` all those things are the internals. We have to define that as a wires ok.

So, you have one more signal which is called as M signal. I will give one more signal which is M signal which is going to decide whether this circuit is going to perform addition or subtraction ok. So, module, so overall block is this 4-Bit_AS adder oblique subtractor ok.

So, what are the different outputs and inputs of these one outputs are S 3 S 2 S 1 S 0 this you can represent with the vector ok. S with the 3 comma 0 is 4 outputs. Another is C out` is the output and what are the inputs we have M, C in under 2 sets of these inputs A with [3: 0], B with [3:0]. This is the initial definition of the inputs and outputs ok.

Then again, individually, write input and output. The final output is S comma 3:0 C out` these are the outputs. Inputs M comma C in comma A, 3 : 0, B, 3 : 0 this is how we can represent the arrays. Now, we have to define some wires. What are the different wires required? You see here what are the connections that are not available outside is here, we require this ok.

I will call this 1 say p 3 p 2 p 1 p 0. So, these 4 wires you have to define then here, we have q 3 q 2 q 1 q 0 ok. Similarly, here, you can call this 1 as r 3 r 2 r 1 r 0 and this C out, we have to define as a wire. And after this NOT gate, you call this some t input this also you have defined as a wire then, Q also another wire.

So, what are the different wires? So, this also you have to make as a ground. So, let us take this one as S we have defined you call this as g 3 g 2 g 1 g 0.

So, what are the total wires? I will write here now. So, these 4 sets g p q r these are the 4 set of the wires that you have to define wire g 3: 0 p 3: 0 q 3: 0 r 3:0 and then, you have C out output of the NOT gate you have defined as t and the overall output you have defined as q. This I have already defined C out this t and this q, I have defined as the wires and then, this q is anyhow the C in is q only ok.

So, these are the things that you have to define as a wires. Now, you have to instantiate first you generate this p 3 p 2 p 1 p 0 using B 3 B 2 B 1 B 0. So, for that you have to write exclusive OR gates, XOR. Output of this 1 is p3 input is M exclusive or with M comma B 3 right this is p 3 is M exclusive OR with B 3. Similarly, p 2 is M exclusive or with B 2 and so on.

Exclusive OR p 2 comma M with B 2, exclusive OR p 1 comma M comma B 1, exclusive OR p 0 M comma B 0. So, what else operations you have to do? So, you have to generate t which is NOT gate output t is the output of the NOT gate input of this one is C in`. Is it

correct C out, output is t is equal to C out bar then, Q is equal to this M and with t and output is q inputs are t and M.

I have generated originated this q's then we have to generate r's by exclusive OR with q and small q's and capital Q, r 3 r 2 r 1 r 0 exclusive OR r 3 is the output Q is 1 input exclusion OR with q 3.

Similarly exclusive OR r 2, Q is the one input q 2 is the other input exclusive OR r 1, Q and small q 1 exclusive or r 0, Q and small q 0. So, this will generate. So, these 4 will be generated r 3 r 2 r 1 r 0 the exclusive or with this ok.

So, we have generated this this C in is anyhow this q ok. Now, this g 3 g 2 g 1 g0 we have to make as 0 that you can do it by using assign. Assign g 3 is equal to 0, assign g 2 is equal to 0, assign g 1 is equal to 0, assign g 0 is equal to 0 these are all initialization.

Now, the actual program becomes. Now, very simple we have to instantiate two such a Ripple-Carry Adders. So, first instantiate Ripple-Carry Adder R C A's. Here, actually for exclusive OR gates you can give some names of these gates also, I have not given here. So, if you want you can give some gate names. I will just start with say G 1 G 2 G 3 G 4 G 5 G 6 G 7 G 8 G 9 G 10 ok.

So, the name that you have given to this R C A in the earlier lecture is 4-Bit_R C A and I am calling the first module as M 1. So, what are the inputs and outputs of this module? We have to just specify only the inputs and outputs this input this input this input and then, this output these outputs. Outputs are q 3 to q 0, C out inputs are p 3 to p 0, A 3 to A 0 and then, M ok.

So, outputs are q 3, C out comma q 3 to 0. These are the outputs and inputs are A 3 : 1 A 3 : 0 is one set of the inputs, then the second set of the inputs will be this this is p. So, this is nothing, but this M ok. So, the second set of the inputs is p 3: 0 and then, M. So, this will perform all the operations that are involved in this ok the outputs are q 3 to q 0.

Now, for the second R C A if you want to instantiate. So, what will be the inputs and outputs. The inputs are r 3 r 2 r 1 r 0 this you have already generated using q 3 q 2 q 1 q 0 ok and then, this is ground and this one is Q itself . this Q you have already generated.

So, second one is 4-Bit_R C A, M 2 block. So, what are the final outputs of this one is C out` is final output that is a fifth bit in case of adder then, we have S 3 is to 0. This is final result then, what are the inputs for these one C in input is we have Q, Q is the C in input. What are the other inputs? This Q is C in input other input is r and g, g's are anyhow 0 ok.

That we have already assigned r's and g's r 3 : 0, g 3 : 0. So, by just instantiating two such Ripple Carry Adders. We will get a 4-Bit Adder/subtractor end module. So, this is how we can instantiate the lower modules to construct the higher modules ok.

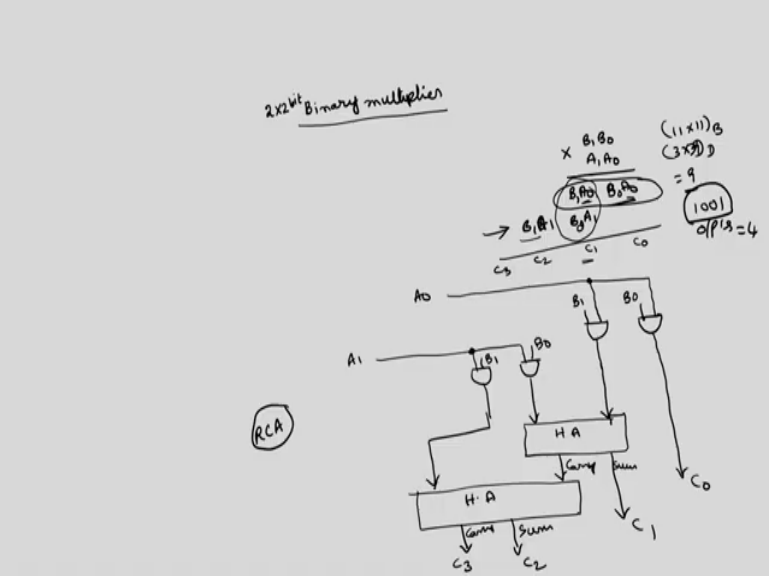
So, we have discussed about the adder subtractor as a common block. So, if I take this A L U, because the most of the digital computer what are the various blocks we have this input block, which is such as the keyboard and mask and we have something like arithmetic logic unit and then, we need some control unit then, some memory this is the overall any processor architecture memory then, we need some output block where you can store the result such as C R T display or L E D 7 segment display ok.

So, we are discussing about this A L U initially. So, I start with the basic operations; such as, arithmetic and logical operations. This will perform arithmetic and logical operations. So, in arithmetic logic operations basically addition subtractions that I have included in a single block that I have discussed.

Now, it can perform the multiplication also, comparison such as a different operation that will be performed. I will discuss some of the operations. Similarly, in the logical we can have shift rotate operations and all. And coming for this control unit, you have something like multiplexers, decoders, encoders, these are all we will include in control unit and you have some flip flops, counters.

So, we will discuss some of these examples of these control units and some examples for this arithmetic and logical operations. So, the next operation is you have discussed the addition, subtraction the next operation is multiplication ok.

(Refer Slide Time: 46:44)



I will just start with a simple multiplication algorithm, binary multiplier. Suppose, if you want to perform 2-bit by 2-bit. So, we have A 1 A0 is one input, B 1 B 0 is another input. So, this normally, I mean conventional like conventional decimal multiplication. You have to multiply A 0 with B 0; A0 with the B 1; A 1 with B 0 A 1 with B 1.

So, this is B0 A 0 is the first bit B 1 A 0 is the second bit. Now, with the A 1 B naught A 1 B 1 A 1 we have to finally add. So, how to implement this this B 0 A 0 means simply AND gate ok. So, you take two AND gates; one input is common which is A 0 here, ok you give this A 0 as 1 input this is A 0, 2 possibilities. We have to give B 0 and yet another place you have to give B 1.

So, these 2's terms will be generated. This as it is the final result if I call this output as C 0 C 1 C 2. We can add one more bit also, because 2-bit by 2-bit multiplication what are the highest number of output that you can be available 1 1 into 1 1 3 into 3 in decimal this is in binary 3 into 3 in decimal which is equal to 9 to represent 9 we require 1 0 0 1. So, outputs are 4.

So, C 3 C 2 C 1 C 0. So, this is simply C 0 and to get C 1. So, we have to take one more set of the gates. So, we take A 1 as another input. So, this will generate A B0this will give as B 1. These 2 terms will be generated by this ok.

So, what you want to do now? B 1 A 0 is here, B 1 A 0 we have to add to B 0 A 1 this is B 0 A 1. These 2 you have to apply to the half adder, because only 2-bit its half adder is enough. So, by adding these 2-bits, you may get 1 sum, sum bit is C 1, carry bit you have to add along with B 1 A 1.

So, the carry of this one, this is sum bit and this is carry bit. So, this you have to add to another half adder whose second bit is A 1 B 1, this is another half adder. So, the sum bit will be C 2 and carry bit will we C 3. This is how we can perform a 2 bit by 2-bit multiplier.

So, if I increase the number of bits then, we have to use Ripple-Carry Adders again. So, I can construct the larger multipliers using Ripple-Carry Adders. So, that I can instantiate this Ripple-Carry Adder only to implement a larger multiplier that we will discuss in the next lecture.

Thank you.