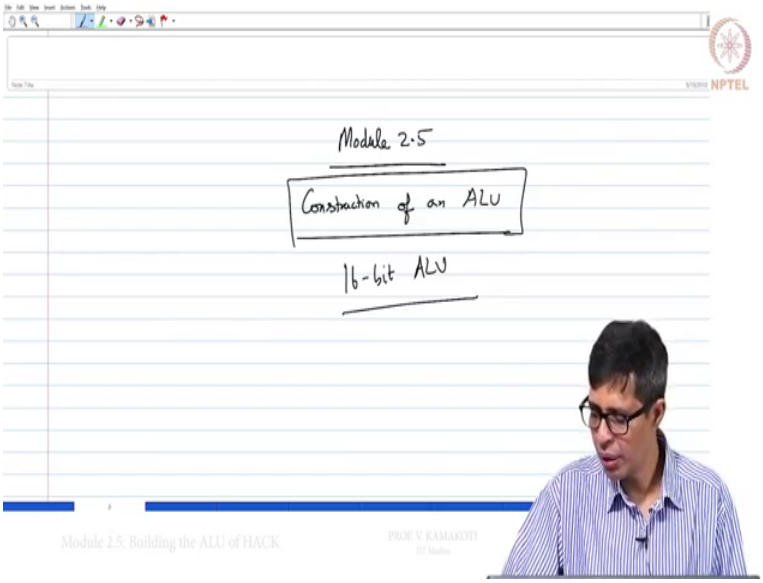
Foundations To Computer Systems Design
Professor V.Kamakoti
Department of Computer Science and Engineering
Indian Institute of Technology, Madras
Module 2.5
Building the ALU of HACK

(Refer Slide Time: 00:20)

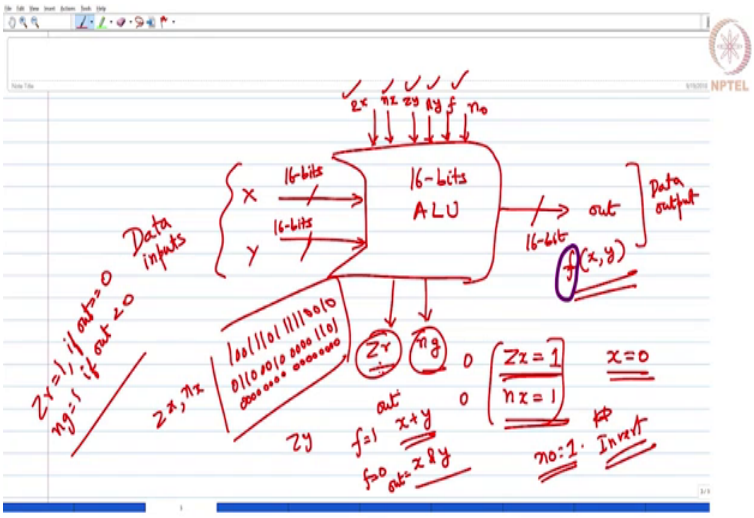


Welcome to module 2.5 and we will be constructing an ALU using whatever we have seen so far we have seen multiplexers we have seen de-multiplexers we have seen four way multiplexer, four way de-multiplexer, four way n-bit de-multiplexer, four way n-bit multiplexer so, so many varieties that we have seen (of) I hope you have done it all those things in your project 01. Now we will use that to basically construct this ALU as a part of this particular module I will basically define what the ALU is and then you will go ahead and basically construct that ALU.

So the ALU is a hierarchical design which will use all those components you have done in the previous project that will be used that will be instantiated and you construct the ALU out of it. So ALU is nothing but the taking out all those components and connecting them together. So you will realize that how easy is to build this ALU provided you have got the basic building blocks, this is typically how normal (you know) even a billion transistor mobile chip is made wherein there are certain building blocks that are already available people will take those the designers

take the building block connect them together to make a large chip. The same thing is what we are going to see here but in a smaller you know way we will be constructing a small 16-bit ALU.

(Refer Slide Time: 2:02)



So I may define the ALU for all, so there are two inputs I call them x and y this are basically 16-bits each that is what we call it as a 16-bit ALU right and so this x and y basically are the data inputs to the ALU and the ALU will outputs again as 16-bit answer which will be some function of x, y what is that function? Some other inputs will determine what is this function f, right? It is the data output of the ALU.

So the ALU when we view an ALU we view it two parts one is the data part another is the control part, the data part will tell you how the data comes and produce the path of it to go out. The control part will tell you what will be the ALU do with that data, the f here is basically determine by the control part of the ALU so this is the data input and this is the data output. What are the control inputs? So there are in this ALU there are six control inputs namely Z x, n x, Z y, n y, f and n o and there are two outputs which are Z r and n g.

Now what will so Z x is 1 right then whatever be your input x but x will be reset to zero so you can give any input x because x is 16-bit right you can give some value on that 16-bit so 1 1 0 0 whatever but the moment I make Z x is 1 x will become zero so whatever you send inside will not go in only zero will go for x right. Similarly when I make n x is equal to 1 whatever input

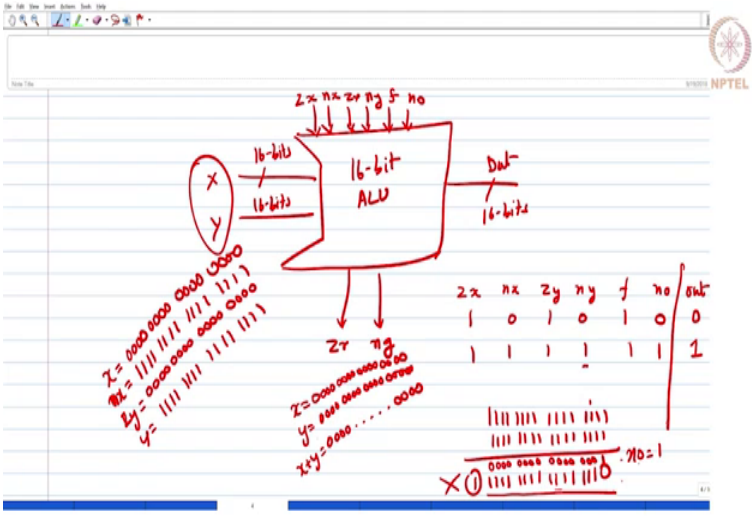
you give to x it will get inverted so if I give say 1100 1110 1111 0010 16-bits for x this will become 0110 0010 0000 1101 so it will just get inverted if the n x is 1, if Z x is 1 all this things will become zero right.

But if both Z x and n x both are zeros right Z x is equal to zero n x is equal to zero x will go as it is here but the moment I make Z x is equal to one this all this inputs will become converted to zero when I make n x is equal to one all this inputs will get inverted. Similarly for Z y so this is for Z x, n x the same thing is true for Z y, n y as you see here, this is Z x, n x this is Z y, n y right then what is f? When f is 1 then x plus y is calculated if is equal to 0 then x and y is calculated.

So when f is equal to 1 out will be x plus y when f is equal to 0 out will be x and y, so that is how f is defined and what is n o? N o is after you compute the output if n o equal to 1 after you compute this f of x, y whether it is x plus y or x and y you invert it. So the normal outpu will get inverted when n o equal to 1, when n o equal to 0 the output will come as it is. So this is how the six control lines will behave right. So the six control lines will tell what you have to do with x or y before you set them as output right and there are two more outputs here namely Z r and n g, Z r will become 1 if output is 0, n g will become 1 if output is less than 0.

So by looking Z and n g you can find whether the output is zero or the output is less than zero or the output is greater than zero, when both are zero when Z r is zero that means n g is not zero output is no zero when n g is also zero that means output is not less than zero so hen Z r and n g both are zeroes we know that the output is greater than zero. So we can quickly as we had seen it 2’s compliment arithmetic w can very quickly find out whether an output is a negative number or a positive number depend on the depending on the we can set up this Z r and n g right. So this is how this entire ALU basically works ok So now we will discuss a bit more about the ALU and tell you how we can use that ALU for different types of operations.

(Refer Slide Time: 8:34)



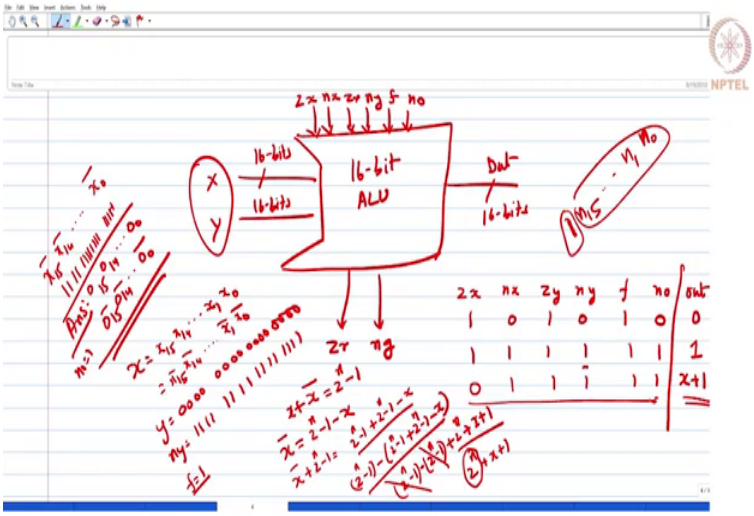
So let me redraw this ALU for you this is the 16-bit ALU and I have the x and y the data coming in 16 bits (16 bits) and then this is output which is 16 bits and then I have Z x, n x, Z y, n y, f n o just six control inputs that we have discussed in the previous slide and then offcourse we have this Z r and zero and not greater than n g right, so we had basically seen so let us just go about.

Suppose I say let us now just draw something like Z x, n x, Z y, n y, f n o suppose I make 1 0 1 0 1 0 what will be it output? So if z x is 1 where x becomes 0, n x is 0 here x doesn't get inverted x is 0 means all the 16 bits are zero so Z y is 1 this also become 0 all the 16 bits become 0 irrespective of whatever you are sending on this x and y here, Z r is 1 y becomes 0 and f is 1, f is 1 means you add so x plus y will again give me 0 (())(10:32) this output whatever is generated is not going to change so the final output would be zero.

Suppose I took 1 1 1 1 1 1 what will happen? Now x see Z x is 0 x is 0 since z y since n x is 1 this gets inverted Z y is 1 so y becomes actually zero then since n y is 0, n y is 1 I am looking at the second here n o is 1 this also gets inverted so a y again becomes 1111 1111 1111 1111 now since f is 1 that means I add so I add 1111 1111 1111 1111 with the same number so this 1 plsu 1 is 0 there will be a carry 1 above, 1 plus 1 plus 1 is 1 again 1 more carry here 1 plus 1 plus 1is 1 so this goes on like this and there will be one carry coming out of this which is 1 because the ALU can handle only 16 bits but extra carry goes off.

So basically we get 1111 1111 1111 1110 now since n o is 1 this output now sine f is 1 we added now n o is 1 this output this now inverted so this will now become because n o is 1 it will now become 0000 0000 0000 0001 so what is the output? Output is 1 right, so like that we can see some very interesting examples.

(Refer Slide Time: 13:01)



So let us say for example I want to increment x, x plus 1 so let us take this so this will be 01 11 11 ok I want to increment x so since Z x is 0 x will come as it is so x will be that x 15, x 14, till x knot x1 x knot and since n x is 1 this will become x 15 bar, x14 bar till x1 bar x knot bar ok.

Since Z y is 1 y is (0000 0000 0000 0000) since n y is 1 it gets inverted so this will become 1111 1111 1111 1111 so what will happen finally that since f is 1 we add this f is 1 we add x 15 bar, x14 bar till x knot bar we add it with 1 1 1 1(1111 1111 1111 1111)ok and we get some answer and this answer so let me say this is o15,o14 to 0 this again be inverted because n o is equal to 1 here so this will become o15 bar, o14 bar till 0 bar, our claim is that this will result to x plus 1. How? Let us take first x comes a when an invert x what is inversion of x? suppose I have x yesterday in the previous module we saw right, x plus x compliment is nothing but 2 power n minus 1.

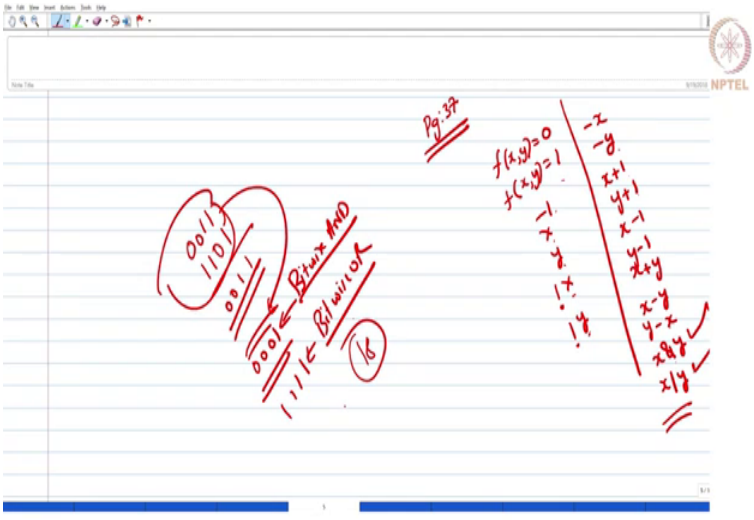
So when you say x compliment this is nothing but 2 power n minus 1 minus x ok right, to this I am adding (1111 1111 1111 1111)what is (1111 1111 1111 1111)? This is nothing but 2 power n

minus 1 so to this x compliment I am adding 2 power n minus 1 which is nothing but 2 power n minus 1 plus 2 power n minus 1 minus x. Again I am whatever answer I am going to get out of this x compliment 2 power n minus 1 that again I am inverting so when I invert this again I will get 2 power n minus 1 minus of 2 power n minus 1 plus 2 power n minus 1 minus x.

So essentially this becomes 2 power n minus 1 plus minus 2 power n minus 1 plus 2 power n plus x plus 1 right. So this is minus so plus 2 power n plus x plus 1 so this two gets cancelled. So finally we have 2 power n plus x plus 1, what is 2 power n? So 2 power n is this bit numbers 16 that is a 16 so we are from a knot to so this s some 16 bit architecture so we are so any number n in the n0, n1 to n 15 right 2 power 16 will be here is a 1 here with (())(17:11) so this 1 automatically will because the computer cannot our computers can only store upto 16-bits this 2 power 16 will be a 17th bit.

We start a 0 we go till a15, 2 power 16 will be the 17th bit right, so this 2 power n so n as the 16th here right so 2 power 16 plus x plus 1 will essentially become x plus 1. So that is how I (()) (17:40) suppose I want to increment x put x here put Z x zero n x 1 Z y 1, n y 1, f1 and n 0 1the answer will be x plus 1 right. So in the center calculation wherever we have used n here this n is 16, this is true for all any n-bit architecture except 32 bit, 17 bit whatever but is also true for 16 bit. So this is how we generate x plus 1 right.

(Refer Slide Time: 18:26)

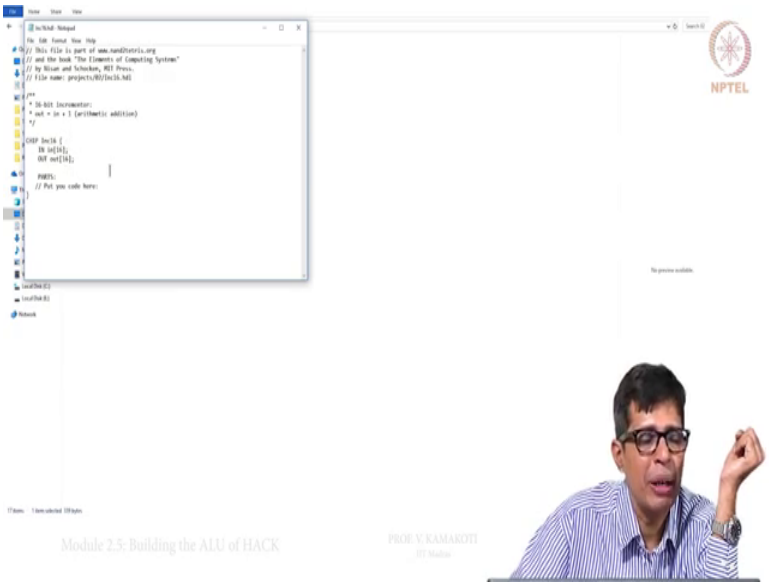


So I will leave it as an a very simple exercise for all of you so you can refer to the book your reference book in page number 37 this available in the reference book but then we need to generate this functions f of xi is equal to 0, f of x y is equal to 1, we have done this two then I want to generate minus 1 x y knot x knot y minus x minus y, x plus 1 which we have done similarly y plus 1, x minus 1, y minus 1 normal x plus y, x minus y, y minus x, x bitwise and y, x bitwise or y.

What is bitwise and this as I mentioned 0 0 0 0 is a 4-bit number, 1 1 1 1 is another number bitwise or let me say make it a little more 0 0 1 1 bitwise and this 1 and 1 is 1(1 and 1 is 1) 0 0 so bitwise under this two is 0 0 1 1 or suppose I make bitwise signed of this two now can essentially become 0 0 0 1 ok. Similarly bitwise OR, the bitwise OR of this 1 OR 1 is 1, 1 OR 0 is1, 1 1 1 1 this is bitwise OR, this is bitwise AND ok. So this is bitwise OR as you see here and between here is bitwise OR. So you need to realize this 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 different functions you need to realize by giving x and y and by adjusting the value of Z x, n x, Z y, n , f and n o by changing this values even basically realize this functions.

So we need to form a table where what would be the combination of Z x, n x, Z y, n y, f and n o those six control inputs so that the function that we are realizing would be this for each one there is a combination by but combination you need to arrive so that you get this answer. We have already seen three such combinations where you have another 15 more to a quote and that is how this whole thing does ok.

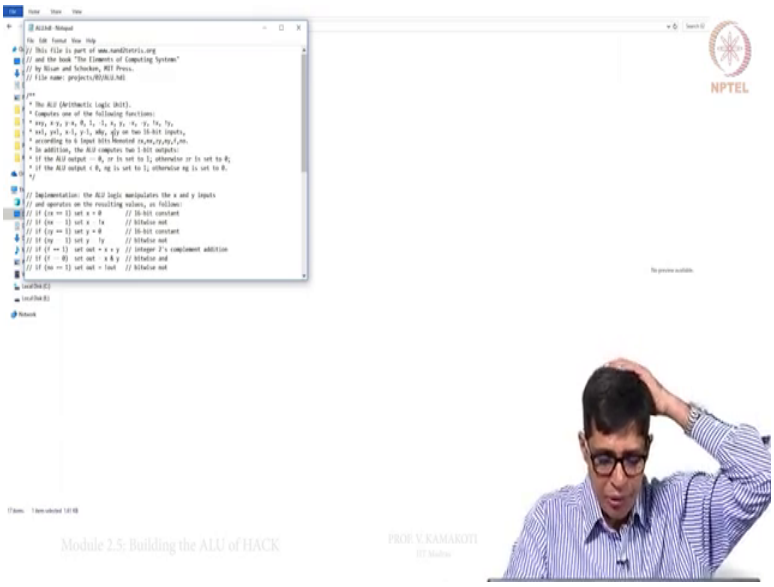
(Refer Slide Time: 21:19)



Now this is the module that we have finished so the project that we are suppose to take just look at this now this is the ALU so there is a full adder but we need to do as a part of this project that is an half that we need to do there is an 16-bit adder that we need to do.

So yesterday we talked about half adder we talked about a full adder and also we talked about a 16-bit adder and right and there is also there is a need for doing 16-bit incrementing right so this is an increment of 16 bit so it should be incremented by one right so this you can use of actually a full adder to increment this (())(22:15) right that can be some other little more intelligent way of doing this also so you can try and do that but nevertheless just use a full 16 bit full adder, 16 bit adder to do this increment.

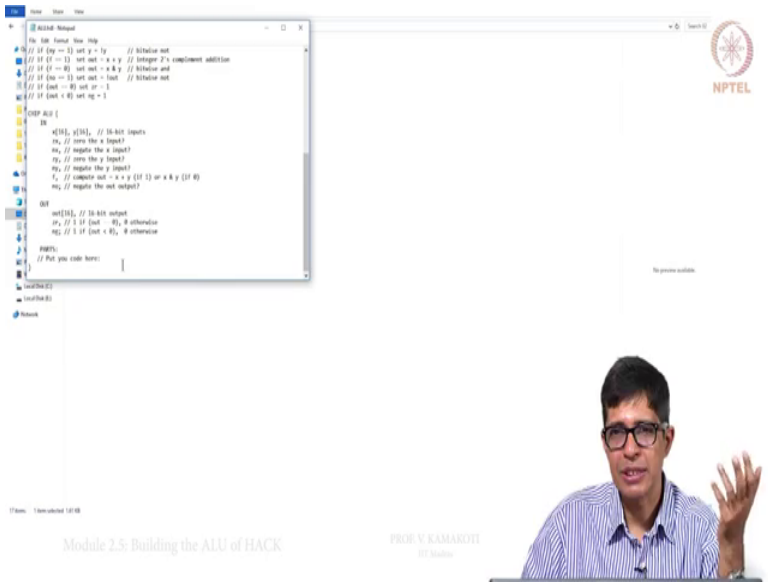
(Refer Slide Time: 22:32)



So we are doing four components here as a part of your project 2 the adder, the first the half adder then the full adder then the actual 16 bit adder then an incrementor so this is a complete hierarchical design the half adder will use the full adder sorry the full adder will use the half adder the 16 bit adder will use the full adder and then the incrementor can also use the 16 bit adder. So once you make this then we need to make this ALU so this is the thing. So the type of functions that we need are given here.

So I need to get x plus y, x minus y, y minus x0, 1 minus 1 x y minus x minus y and all this things we need to realize right. So we have realized three such functions but there are fifteen more such functions that you need to realize. Table for this is available in the book at page number 37 so if you have the book you can refer to that book others have to think and get out with this some of them are not immediately (())(23:45)but you have to think to get this functions in place right we have done for example we have done x plus 1 is actually not very trivial but we did then show how x plus 1 works, so like that we need to do for this functions here and this functions are mixed up in your project 2 this ALU dot HTL file.

(Refer Slide Time: 24:09)



So then we loop so the moment is finish an half adder is (())(24:12) you put your code here immediately you can go and load the script here dot txt script here and there is already a compare script so it will save as a of passing whether the design is correct or not. Similarly then with that you can build a full adder again you can do this comparison there is a compare and they are test there is a script for testing the full adder then there is a script for testing a add 16 there is a script for testing your incrementor there is also a test for testing ALU so you can keep building.

So another small point is when we are constructing the ALU so basically ofr you to realize this functions all this functions could be realized you just using the incrementor, the adder, the full adder, the and the MUX's and De-MUX's multi bit gates and other things that we have already constructed as part of a project 01 and till now project 02. So we have constructed lot of building blocks in project 01 and now we are constructed this four circuits namely full adder, incrementor, half adder and the half adder, full adder, adder and incrementor using this four plus whatever we have done on project 01 we can those are the basic building blocks for building this ALU right.

So with this we complete module 2 any doubts you have please put on the forum and we can basically take it forward from that, thank you.