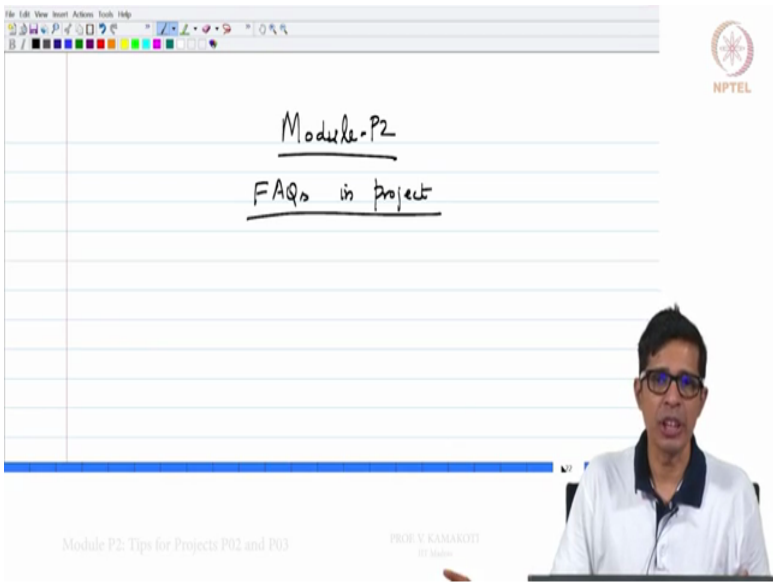
Foundations to Computer System Design

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

Module P2
Tips for Projects P02 and P03

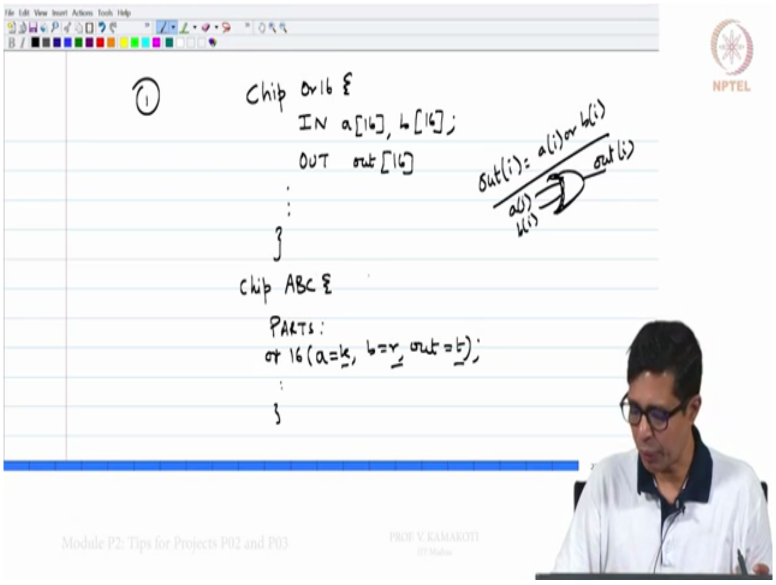
(Refer Slide Time: 0:16)



So welcome to module P2, so this is basically done to address some of the frequently asked questions in the project, so that this helps you to finish the project faster, so again I insist that this course for it to be fruitful to you, you have to do the projects I hope you are doing it well and you are in the minutes of finishing your project too.

So we will give you certain inputs that will be extremely useful for you to finish the project quickly and whatever I am going to talk today will make lot more sense if you are actually tried and there have been having some issues and also please do write all your issues on the discussion forum, so that you get quick responses do not wait for you know the modules to complete you can keep progressing well you can used the book, you can used the online source and you can actually make some progress earlier, so that when you actually see through the lecture it becomes much more fruitful for you.

(Refer Slide Time: 01:21)

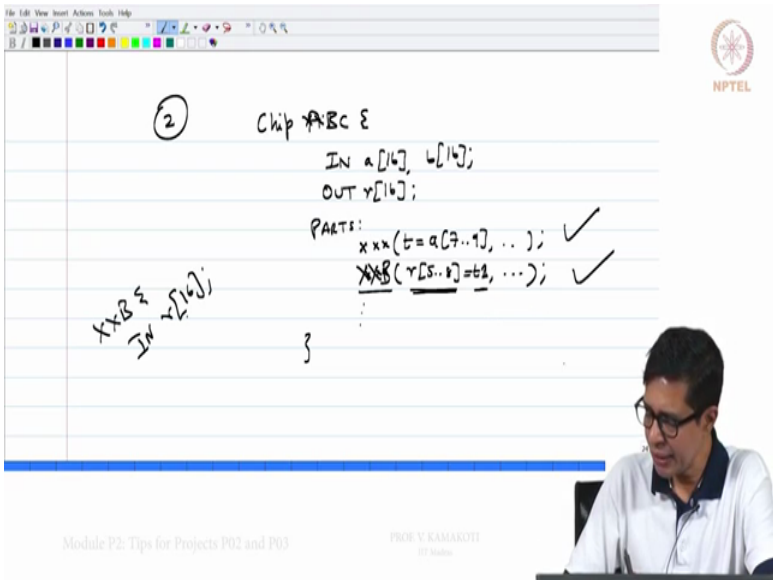


```
①      Chip Or16 {
           IN a[16], b[16];
           OUT  out[16]
           :
           :
       }
       Chip ABC {
         PARTS:
         or 16(a=k, b=r, out=t);
           :
       }
```

Now module P2 is some facts in projects that you have done so far, so let us take this Or 16 is a chip that you are designed which will take 16 bits and 16 combinations of bits a, b and it will give you a 16 bit output which will be out i is equal to a i or b i, i (())(01:38) from 0 to 15 so that is Or 16. So it is takes two vectors a and b of 16 bits in length and (intel) generates an output vector which is again 16 bits in length out i is a i or b i, ok.

Now, in some other chip a, b, c we are using the same Or 16 where in ABC a equal to k, b equal to r, out equal to t let us assume that k, r and t are internal variables and part of the input and output, we have already seen in the module P1 that k, r and t automatically assumes to be a 16 bit vector right, they are automatically assigned as a 16 bit vector, so you need that explicitly say that k is a 16 bit vector since it is actually assigned to a which is a 16 bit vector k becomes a 16 bit vector similarly r becomes an 16 bit vector (Or) same as t where k, r, t are internal variables not described as part of the in and out, right. So this we are seen in project P1.

(Refer Slide Time: 02:58)



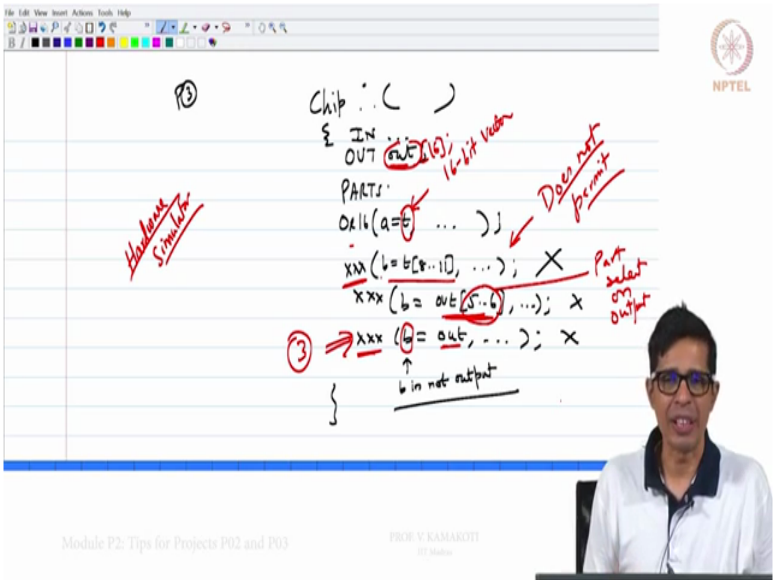
```
(2)    Chip ABC {
           IN a[16], b[16];
           OUT r[16];
           PARTS:
               xxx(t = a[7..9], .. );      ✓
               XXB( r[5..8] = t1, ...);    ✓
               :
       }

XXB {
  IN r[16];
```

Now, let us see some very interesting things now suppose I defined a chip ABC where A and B are inputs of 16 bits vectors and out is another r is an output which is 16 bit vector. I instantiate several chips inside so let us say xxx some chip and t is an input variable to that so I can say t equal to a 7 to 9 right, so I can take part of the input and assigned as input to that module. Similarly I can say r 5 to 8 is equal to say some t 1, right.

So this is some input of so let us take this xxB some XXB or something like that so XXB has an input which we call it as r some 16. Now, I can assigned t 1 any variable as part of that input variable, so both of these are correct so what we are saying here is that if I have an input variable and I am instantiating a chip inside I can take part of that input variable and assigned it to that t right, similarly (if I) if there is an input of another chip which is think which act as a vector I can take I can assigned part of that vector to some variable t 1 here, so t 1 essentially (is a 3) should be a 3 bit 4 bit variable here, ok. So both of these are correct, so these two will be supported.

(Refer Slide Time: 04:49)



```
P3

Chip : ( )
{ IN ...
  OUT out[16];
  PARTS:
  Or16(a=t ... );
  xxx(b=t[8..11], ...);   X
  xxx(b= out[5..6], ...);  X
  xxx(b= out, ...);   X
}
```

Now, let us see what will not be supported, for example I have a chip in out or 16 a equal to t, now so t actually now becomes a 16 bit vector, right. Now t is an internal variable speaking a 16 bit vector, your simulator or hardware simulator which we call it your hardware simulator that we are working on will not is allow this. I cannot taken internal variable and then used part select on that, right.

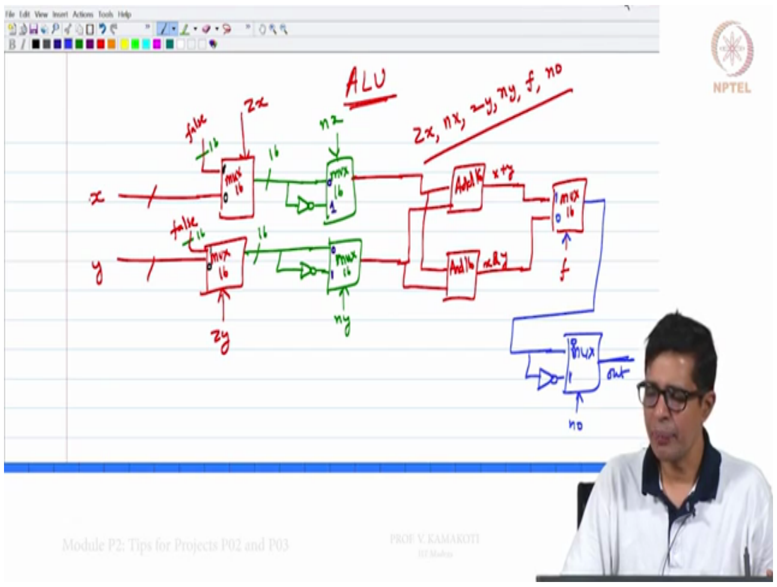
So I am instantiating some other chips x swap b is part of this internal variable if it is also part of input output X but if it is a part of internal variable I cannot split it like this, so this your hardware simulator does not permit, ok. So you should be careful while doing this. Similarly if I have an out variable out ok, let us take this is out 16 for example I cannot make part of that output b equal to something right, first and foremost so I cannot do a parts select on the output and more than that if suppose let us take this case which I am marking here if b is not the output variable of some module xxx b is in input variable of that module then I cannot assign out to that input variable.

So the output variable cannot become and input to some other chip which is instantiated within there right, please note that in this case xxx the third case I am talking of here xxx b is an input variable to xxx b is not an output variable of that to that I cannot assign the output variable of the chip, the output variable of the chip is out that I cannot assign to an input variable of some other chip.

So these 3 are very important so to summarize the hardware simulator does not support a parts select of an internal variable in this case t is an internal variable since it has been used to it a equal to t on Or 16 (a) t becomes a 16 bit vector I cannot used the path select of an internal variable assigned to something, I cannot do a path select of an output variable in this case out I cannot do that and similarly I cannot assign a path output variable of the chip to in input variable of some other chip that I am instantiating here, right.

So these 3 are important conditions that you should basically look into very carefully when you are designing your hardware, so right. These are some limitations of the or I do not know we cannot call it stimulation these are all some of the rules are constraints that are imposed by your hardware simulator. Never classify follow these rules your actual design becomes much more readable or much more verifiable, right.

(Refer Slide Time: 08:07)



So, now let us go to the next part so you would have been designing the ALU, so just to summarize ALU had (n x sorry n a z, n x let me just do that if that is z x sorry) is Z x, n x, z y, n y, f, n o six input control lines right, so this has been explained well in the previous module. So let us take so the ALU get us 216 bit vectors namely x and y how do we work on this so let us put this as a mux 16 so we just put a mux 16 we also put another mux 16 here and basically give false, false means zero right, so false and the control select this basically is Z x and Z y right. So if you give Z x the output will be zero if this is so this is 1 and this is 0 similarly this is 1 and 0, ok. So if Z x is 1 output is zero otherwise it is x similarly Z y is 1 output 0 otherwise it is x right.

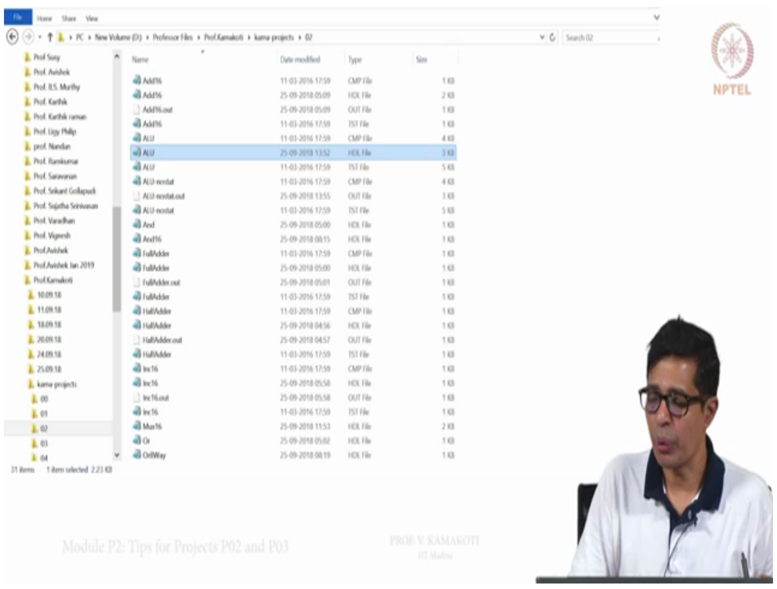
Now we proceed to the next stage so this output we take we inverted similarly this output AND inverted not the there are 16 bit lines all of them are 16 bit, this also 16 bit and this is also 16 bit, now we basically give again to a mux 16 and we give here n x and we give here (n y) mux 16 and we give here n y, so this is again this is 0, 1and this will be 0 and 1. Now look what is happening so if n x is 0 then whatever comes out of (these mux 16) this mux 16 will come out here if n x is 1 whatever come out these mux 16 get us inverted and you get it here, so that is what right.

So this is the x you have defined this is the y you have defined that x will become not x if n x is 1, y will become not y when n y is 1 otherwise it passes on so this is the next thing then you basically have not adder the 16 bit adder and you will have the 16 bit and you basically give these two inputs and also these two inputs so x and y you give it to both. So you will get an answer here x plus y and you will get x and y here, right.

Now again give it to your mux 16 and here the input will be f and this is 1 and this is 0, if f is 1 it is x plus y, f is 0 it is x and y I will again take this input and give it to another mux where you put n o this is 0 and this is 1 and these is your final output, ok. So now you actually see how the control signals are playing a role and how the data flows, so note that the control signals basically give the data through that path, ok so which data should go where the control signals basically to this, so all the control signals are control to the multiplexes here right and so multiplex are actually route which data has to go which is decide at every stage what is the data that needs to be proceed it.

So z x, z y is this x whether x should go or 0 should go, y should go or 0 should go after that the n x, n y decides should I invert that value of x or not then it is getting adder than that then this mux will decides should I give the adders output or the AND output and then finally the last mux decides whether should I invert the output or not invert the output, so this is how the ALU is constructed please note that this is essentially a very modular design we already done mux 16, we have already done add 16, we have done AND 16 just we have to write a small code which will be some five to ten lines to basically do these whole project and it will work,

(Refer Slide Time: 14:01)



PC › New Volume (D:) › Professor Files › Prof.Kamakoti › kama projects › 02

Prof Suny
Prof. Avishek
Prof. B.S. Murthy
Prof. Karthik
Prof. Karthik raman
Prof. Ligy Philip
prof. Nandan
Prof. Ramkumar
Prof. Saravanan
Prof. Srikant Gollapudi
Prof. Sujatha Srinivasan
Prof. Varadhan
Prof. Vignesh
Prof.Avishek
Prof.Avishek Jan 2019
Prof.Kamakoti
10.09.18
11.09.18
18.09.18
20.09.18
24.09.18
25.09.18
kama projects
00
01
02
03
04

Name	Date modified	Type	Size
Add16	11-03-2016 17:59	CMP File	1 KB
Add16	25-09-2018 05:09	HDL File	2 KB
Add16.out	25-09-2018 05:09	OUT File	1 KB
Add16	11-03-2016 17:59	TST File	1 KB
ALU	11-03-2016 17:59	CMP File	4 KB
ALU	25-09-2018 13:52	HDL File	3 KB
ALU	11-03-2016 17:59	TST File	5 KB
ALU-nostat	11-03-2016 17:59	CMP File	4 KB
ALU-nostat.out	25-09-2018 13:55	OUT File	3 KB
ALU-nostat	11-03-2016 17:59	TST File	5 KB
And	25-09-2018 05:00	HDL File	1 KB
And16	25-09-2018 08:15	HDL File	1 KB
FullAdder	11-03-2016 17:59	CMP File	1 KB
FullAdder	25-09-2018 05:00	HDL File	1 KB
FullAdder.out	25-09-2018 05:01	OUT File	1 KB
FullAdder	11-03-2016 17:59	TST File	1 KB
HalfAdder	11-03-2016 17:59	CMP File	1 KB
HalfAdder	25-09-2018 04:56	HDL File	1 KB
HalfAdder.out	25-09-2018 04:57	OUT File	1 KB
HalfAdder	11-03-2016 17:59	TST File	1 KB
Inc16	11-03-2016 17:59	CMP File	1 KB
Inc16	25-09-2018 05:58	HDL File	1 KB
Inc16.out	25-09-2018 05:58	OUT File	1 KB
Inc16	11-03-2016 17:59	TST File	1 KB
Mux16	25-09-2018 11:53	HDL File	2 KB
Or	25-09-2018 05:02	HDL File	1 KB
Or8Way	25-09-2018 08:19	HDL File	1 KB

31 items   1 item selected 2.23 KB

Module P2: Tips for Projects P02 and P03

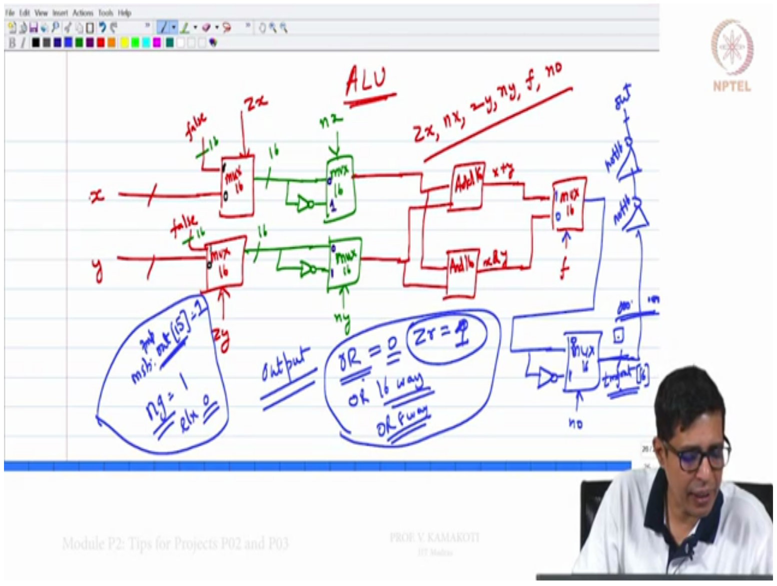
PROF. V. KAMAKOTI
IIT Madras

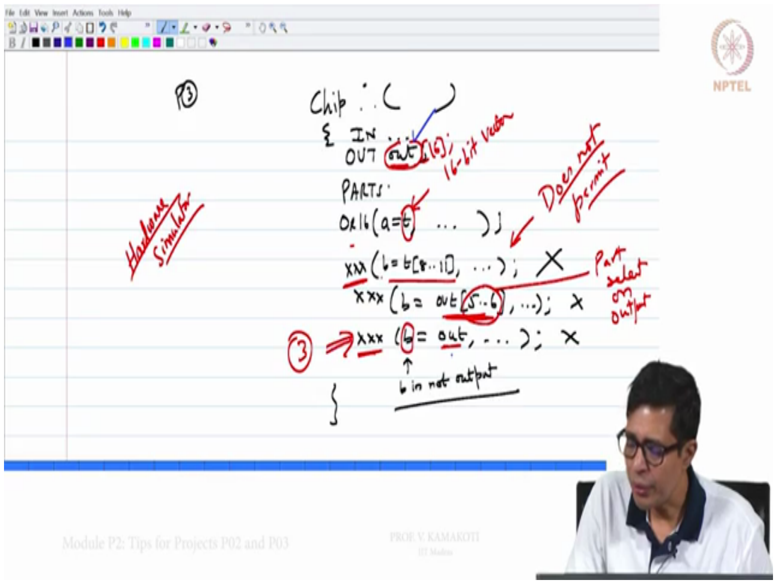
Now, once (you) we go to ALU dot hdl and basically you know we can go to ALU dot hdl and basically you can go ahead and look at this file, so this going to be a very simple small file so these is the total thing that we need to do, ok. So this is a small file in that so small set of code that we need to write I am just leaving it as an exercise for you to do then what we need to do is after you finish this ALU then of course there is a text file and there is a compare file, the text has two parts ALU know status this is one text where there are two flags that come out namely your is z r and n g, two flags that come out whether that say that answer is zero or the answer is less than zero, answer is negative.

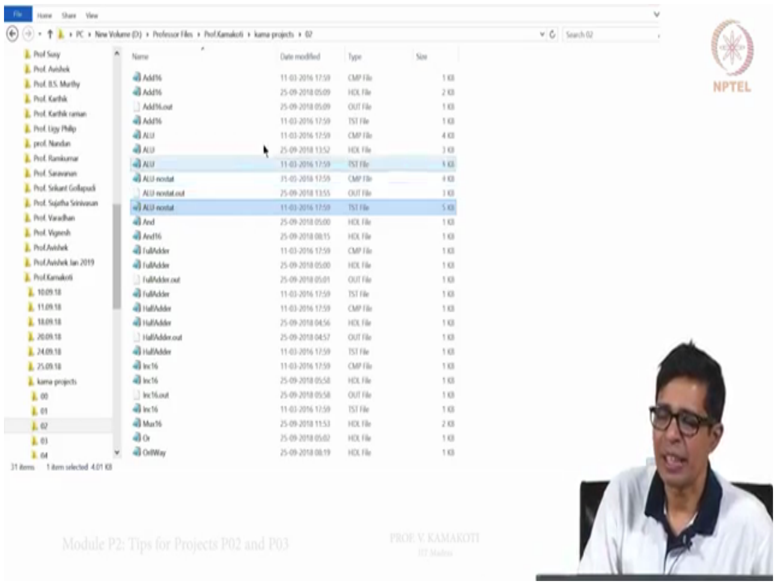
So what I have described so far we have not coded the flags right, the status flag so basically you can used this ALU knows that to test your ALU so far whether the status flag is not output then you can used the normal ALU txt file which will include the status flag also this will include the status flag so you can do the ALU in two stages, the first ALU is do not include the status flag then you can do the you can include the status flag circuitry and then you can do the this ALU dot txt before you do ALU no stat dot txt.

So there are two text files for testing the ALU, one with the status flag and one without the status flag, first do without the status flag to see that it works properly then do with the status flag after your code that part.

(Refer Slide Time: 16:08)






New Volume (D:) › Professor Files › Prof.Kamakoti › kama-projects › 02

Name	Date modified	Type	Size
Add16	11-03-2016 17:59	CMP File	1 KB
Add16	25-09-2018 05:09	HDL File	2 KB
Add16.out	25-09-2018 05:09	OUT File	1 KB
Add16	11-03-2016 17:59	TST File	1 KB
ALU	11-03-2016 17:59	CMP File	4 KB
ALU	25-09-2018 13:52	HDL File	3 KB
ALU	11-03-2016 17:59	TST File	5 KB
ALU-nostat	11-03-2016 17:59	CMP File	4 KB
ALU-nostat.out	25-09-2018 13:55	OUT File	3 KB
ALU-nostat	11-03-2016 17:59	TST File	5 KB
And	25-09-2018 05:00	HDL File	1 KB
And16	25-09-2018 08:15	HDL File	1 KB
FullAdder	11-03-2016 17:59	CMP File	1 KB
FullAdder	25-09-2018 05:00	HDL File	1 KB
FullAdder.out	25-09-2018 05:01	OUT File	1 KB
FullAdder	11-03-2016 17:59	TST File	1 KB
HalfAdder	11-03-2016 17:59	CMP File	1 KB
HalfAdder	25-09-2018 04:56	HDL File	1 KB
HalfAdder.out	25-09-2018 04:57	OUT File	1 KB
HalfAdder	11-03-2016 17:59	TST File	1 KB
Inc16	11-03-2016 17:59	CMP File	1 KB
Inc16	25-09-2018 05:58	HDL File	1 KB
Inc16.out	25-09-2018 05:58	OUT File	1 KB
Inc16	11-03-2016 17:59	TST File	1 KB
Mux16	25-09-2018 11:53	HDL File	2 KB
Or	25-09-2018 05:02	HDL File	1 KB
Or8Way	25-09-2018 08:19	HDL File	1 KB

31 items 1 item selected 4.01 KB

So let me just go and you know show that parts so how do you generates the status flag, one thing is you can take the output and if you take the output and AND all of them sorry if you take the output and OR all of them if the OR is 0 that means is Z r should be 1, right. Basically you can OR all the with is or equal to 0 then Z r will become 1 so this is one part of the story.

So this output we take and we OR all the bits of these outputs and then you get Z r equal to 1, if that OR is 0 that means none of the bits are 1 output is 000000 16 th times so it will become 0. So (you cannot) have to do an OR 16 th way, right you could have done an OR 8 th way (we have) so we can used to OR 8 th ways to do this OR 16 bit. The second thing is negative n g, the n g flag f to take the most significant bit of the this out and if that most significant bit m s b of out which is out 15, if this out 15 is 1 then n g is 1 else it is 0, ok. So this is how you basically find the n g flag and this is how you basically find the Z r flag, right.

Now the challenge that you will be facing here is that you cannot take this output and input to any other module as just I have told you in the previous screens like if this is an output that output variable cannot be given assign to ant input variable of any chip that I am instantiating. So you cannot just go and say that you can take this output and give it to your module which basically goes and test some part and come out, ok.

So what we need to do is we may have to go and make it as a temp out, ok and this temp out so you may have to inverted two times to get out, ok. Now this temp out actually becomes an internal variable and this internal variable I can take it out to check and to assign temp out equal to out, right. So there is no buffer that we have (that is) which will take this temp out and give out an out.

So I have to invert I have to put two not 16 here basically to get an out here, that is the only way by which I can assign out to temp out, right. That is nothing like a buffer which will just as carry this values and assign it, out and temp out are the same but there is no way by which I could express it using our built in chips, OR the chips that we have design, so we have to put two not is to get them output there, right. Then this temp out you can tap to basically start using to basically get your n g and your Z r, ok right.

So this temp out can be used on an OR 8 th way, 2 OR 8 th way is and we can do but please also note that this temp out being an internal variable I cannot split it, this temp out is a 16 th bit variable, right. So (this is a 16 th bit) this is a mux 16 so this are 16 th bit variable again I cannot split it as 0 to 7 and 8 to 15 that is not possible here because as I told in one of the rules the internal variable we cannot do a path select on a internal variable, so this temp out has to be treated together and we are only an OR 8 th way we do not have an OR 16 th way, so that also we need to keep in mind, right and the secondly the I cannot take a path select of a internal variables so I cannot even get this temp out of 15 which I if it is 1 then n g is 1, so I cannot get temp out of 15, so essentially I have to do something to I have to generate two other modules or chips to basically handle this.

So there will be small challenge here I will leave it as an exercise for the reader to basically get your n g and Z r correctly tested. After you code that n g and Z r into your ALU dot hdl chip then you can used your ALU dot txt currently without this n g and Z r module you can used the script ALU no stat and if with this we can actually used the script this is ALU no stat and once you code the for the Z r and the n g flag then you can used your ALU dot txt still then used to ALU no stat dot txt.

So these are some of the things which will help you to quickly finish your project ALU, right and I hope this is going to be useful if you have any other doubts please do put in the discussion forum and we will respond to it, it is very important that you finish this ALU before you start looking into the third module, all the very best I hope you will your is enjoying this course, thank you.