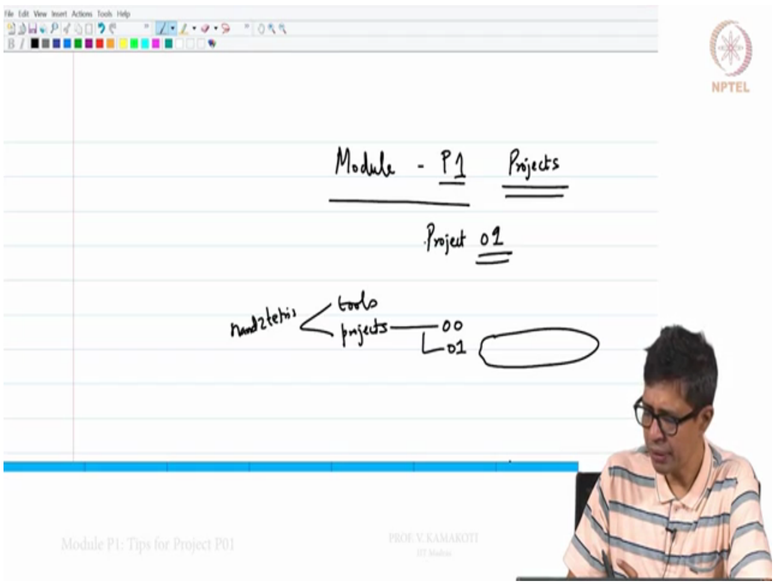
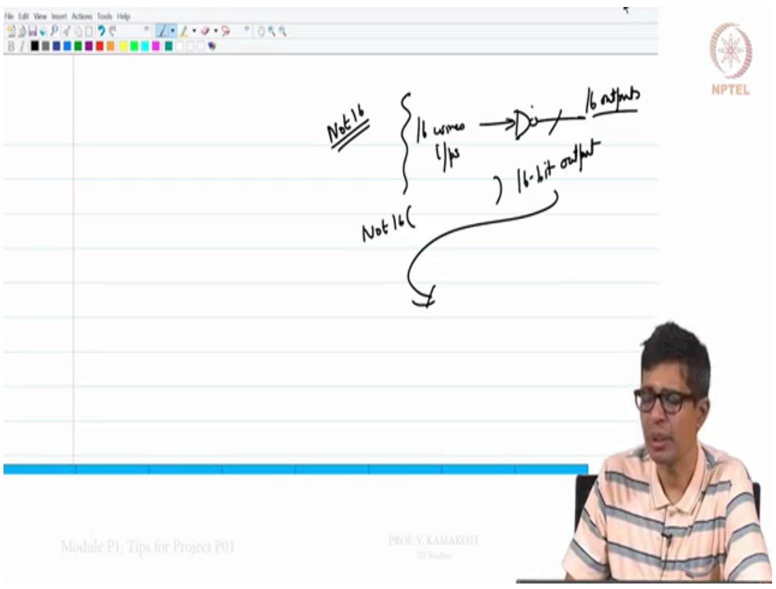
Foundations to Computer Systems Design.
Professor V. Kamakoti.
Department of Computer Science and Engineering.
Indian Institute of Technology, Madras.
Module P1.
Tips for Project P01.

(Refer Slide Time: 0:19)



So, welcome to module P1. This is nothing but we are going to have a set of modules, specifically dedicated for projects. So, we will explain certain concepts that will help you to quickly finish the projects as a set of commander these modules P1, P2, P3, P4. As and when we feel that there is necessary for giving some extra inputs are specifically for the project, we will basically do it here. So this P1, basically were trying to give you some inputs related to your project 01.

(Refer Slide Time: 1:50)

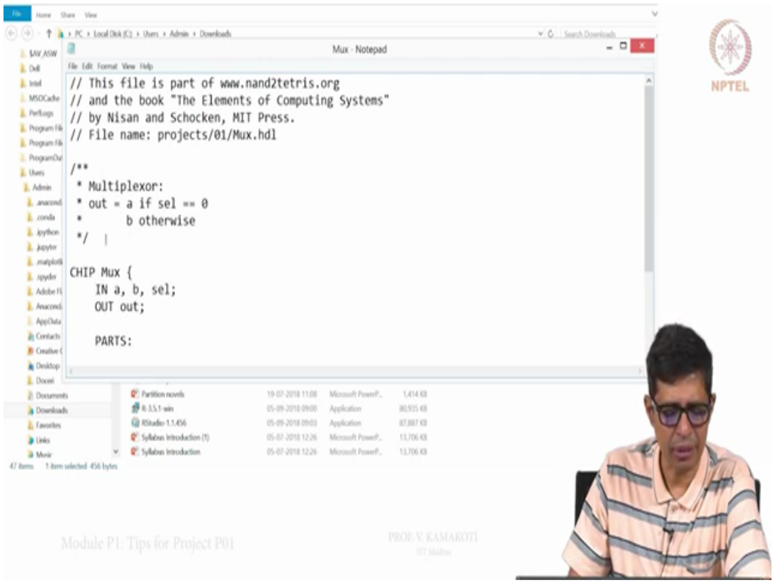


So when you had downloaded NAND to Tetris, you would have got to directories, one is called tools, another is called projects, both will be in a directory called NAND to Tetris. Now we can go into projects, there you will find 00, which is just a file, then 01, now, you must be doing the 01 now. And we will explain certain concepts of 01, certain concepts which will help you to copy 01 faster, right.

So one of the interesting thing is that as we are doing 01, you would have realised that there are things like you are having Not 16, where you are handling 16 wires at a time and you are getting 16 inputs at a time given to the virus, each of them Noted and you are going to give get 16 outputs, like that multibit gates, multiway gets. So now we are seeing, we try to process a vector of inputs. Not a single wire but a collection of wires, vector of wires, right. So that is very very important.

So how do we transfer 1 vector of wire to another gate? Let us take this Not 16 is giving me 16-bit output, right because there will be 16 wires. The 16 bit again I have to pipe it some other gates. So how are you going to all this passing on, so that we will do with the help of 3 examples, very quick examples, which is also part of your project 01.

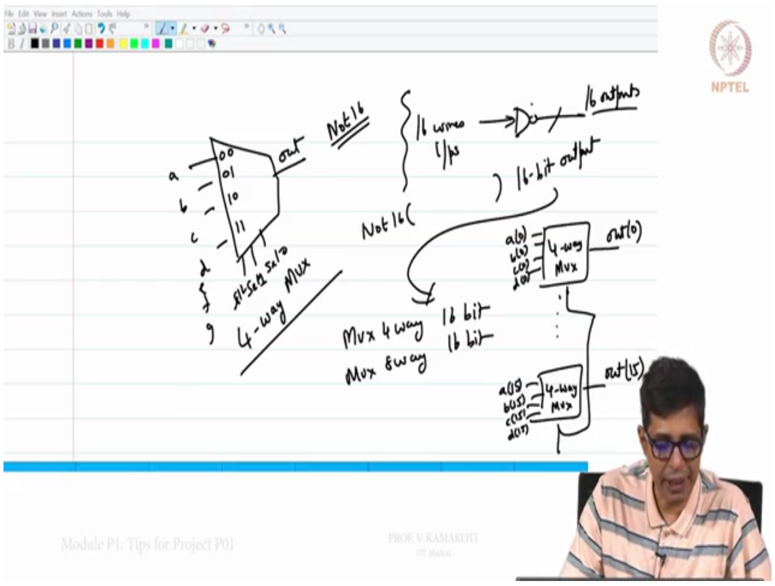
(Refer Slide Time: 3:10)

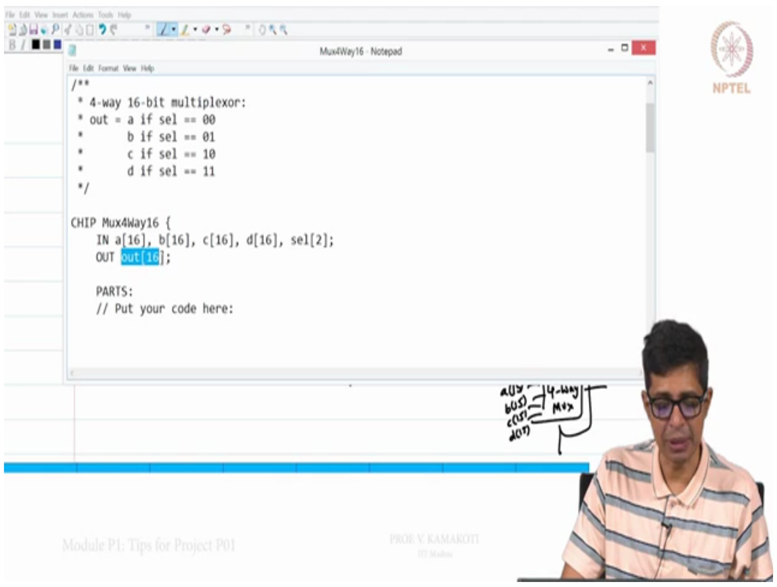


```
// This file is part of www.nand2tetris.org
// and the book "The Elements of Computing Systems"
// by Nisan and Schocken, MIT Press.
// File name: projects/01/Mux.hdl

/**
 * Multiplexor:
 * out = a if sel == 0
 *       b otherwise
 */

CHIP Mux {
    IN a, b, sel;
    OUT out;

    PARTS:
```

Right, see the 1st one, this is mux, right, how do you write a mux. Mux is given A, B, given A and B and select, we basically have to, if select is 0, your output A, otherwise you output B. So this is written, so mux is done, I wish there are 3 inputs A, B and select and out is the output. So if select a 0, output is A, else output is B. Now, using this we want to make a mux 4 way 16. 4 way 16 means I want to basically construct a 4 way mux to start with. What is a 4 way mux?

(Refer Slide Time: 3:58)





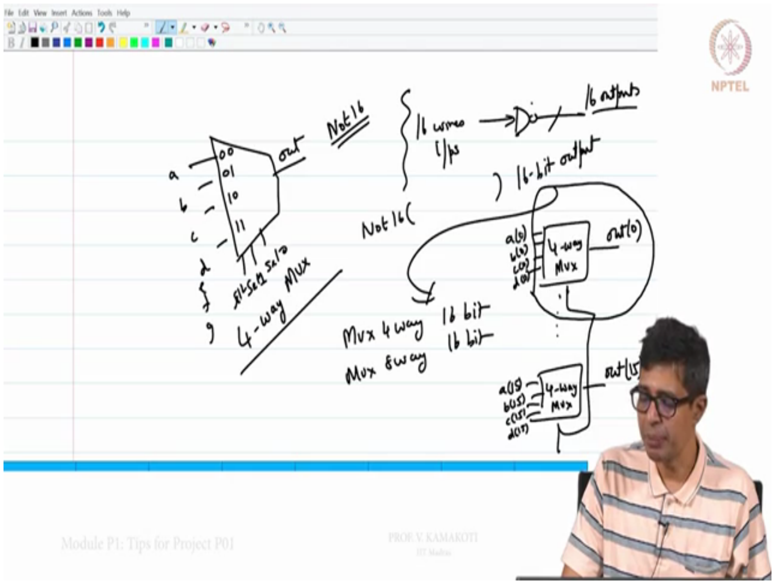
```
/**
 * 4-way 16-bit multiplexor:
 * out = a if sel == 00
 *       b if sel == 01
 *       c if sel == 10
 *       d if sel == 11
 */

CHIP Mux4Way16 {
    IN a[16], b[16], c[16], d[16], sel[2];
    OUT out[16];

    PARTS:
    // Put your code here:
```

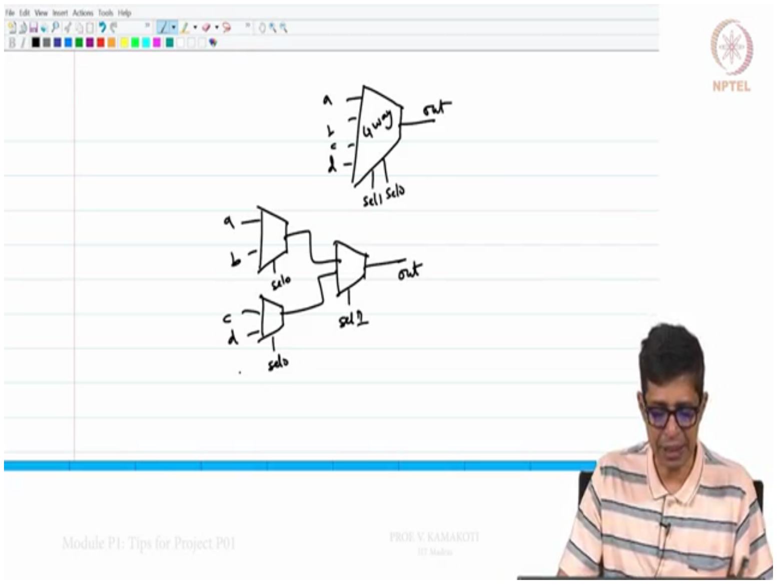
4 way mux in is that I have A, B, C, D and basically I have 2 select inputs, select 0 and select 1 and this is out. If these 2 bits and 00 I get this 01, 10, 11. So this is 00 out equal to A, 01 out equal to B, out equal to C, out equal to D, right. So this is what we call as a 4 way mux. Right, now I am now talking of a mux which is 4 way and 16-bit. Essentially this is one structure, let me call it, this is a 4 way mux that takes 4 inputs and gives me an output. So I will replicate it 16 times, that is mux 4 way 16-bit. Right.

So I will give A 0, A1, A2, A3 here, sorry, A0, B0 I give A0, B0, C0, D0 here, like I give A 15, B 15, C15, D 15 here and the select is common to all and I get out 0 to also 15, we have explain that earlier. So, this is a mux 4 way 16-bit. Now I could also have 8 way 16-bit. I will have to sel 0, sel 1, sel 2 and then I will have 8 selections here, so A, B, C, D, E, F, G, H, depending on 0000, 0101… 0111 till 8. So, now what we will see here is mux 4 way 16, right.

Now how do we do this 4 way 16-bit multiplexer? We just do a 4 way one and repeat it. Now, essentially what we are doing here is that mux A equal to A0, B equal to B0, sel equal to sel 0, out is T01, A equal to C0, B equal to D0, sel equal to sel 0, out equal to T 02 and so on. So what we want to demonstrate here is that you see sel is a 2 bit input that is given to you, sel is a 2 bit input given to you. A, B, C, D are 16-bit inputs that are given to you and out is a 16-bit input.

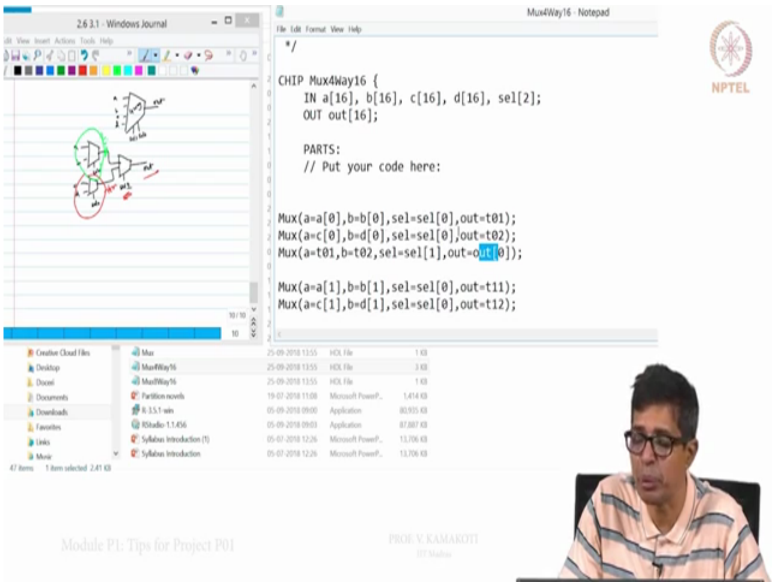
(Refer Slide Time: 7:09)





Now as you see, I want to basically do this mux as you see here. This is a mux, which basically takes A0, B0, C0, D0, sel 0, sel 1 and basically gives me out 0. How do we do that, this is as follows. So what we need to understand here, we have already seen how to do a 4 way mux from a two-way mux but what we see is how we are going to basically pass the inputs. So, I am putting 3 muxes here, so suppose I want to do this A, B, C, D, sel 1, sel 0, out, this is nothing but do A, B do C, D, both with sel 0 and then take this output, do it with sel 1 and you get out. So, now this has been shown earlier. So this 4 way mux can be realised using 3 two-way muxes and that is precisely where doing here.

(Refer Slide Time: 8:29)



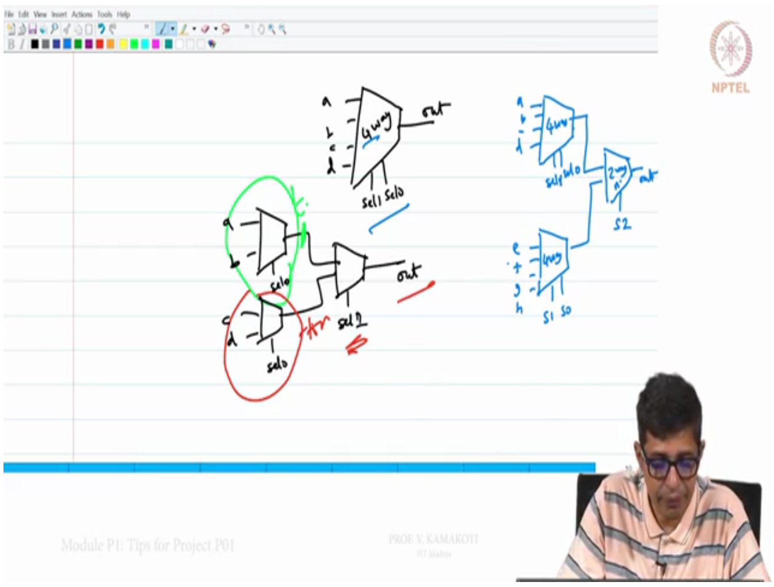
```
 */

CHIP Mux4Way16 {
    IN a[16], b[16], c[16], d[16], sel[2];
    OUT out[16];

    PARTS:
    // Put your code here:


Mux(a=a[0],b=b[0],sel=sel[0],out=t01);
Mux(a=c[0],b=d[0],sel=sel[0],out=t02);
Mux(a=t01,b=t02,sel=sel[1],out=out[0]);

Mux(a=a[1],b=b[1],sel=sel[0],out=t11);
Mux(a=c[1],b=d[1],sel=sel[0],out=t12);
```

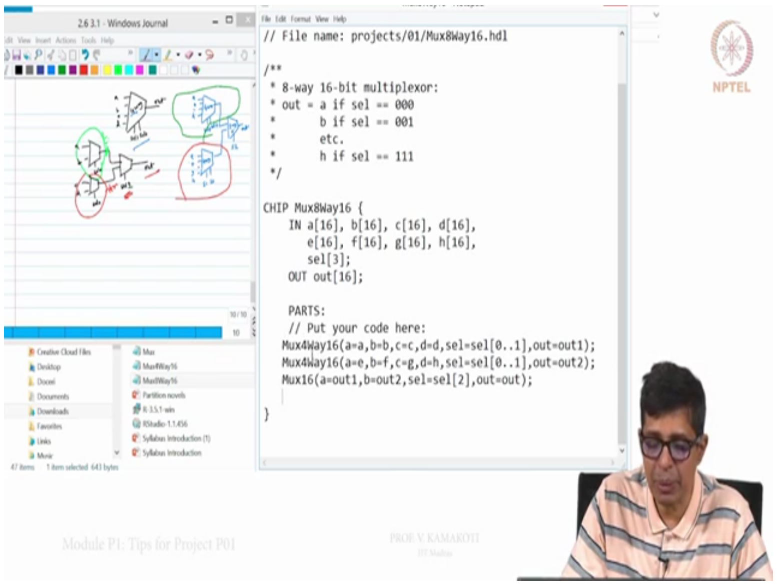
As you see here, precisely what we are doing here. So as you see at this part, so, mux A, the 1st one will take A0, B0 and sel 0, so the 1st line that we are seeing here is corresponding to this particular mux that we need to say here. Right, A 0, B0, sel 0, out on some T01, this wire which I am marking in green is T01. The 2nd mux you see here, the 2nd mux you see here takes C0, D0, again sel 0 and out is T02. The 3rd mux you see here, it takes T01 and T02 as inputs and sel 1 as the select, as you see here, sel 1 as the select and out is here.

So this particular 4 way mux we are now replicating it 15 times. What you need to understand here is that for the 1st mux A0 is the input, for the 2nd mux, A0, B0, C0, D and sel 0, sel 1. So I can take, so I have declared a 16-bit input here and 2 bit select here, many 16-bit inputs and 2 bit select. I can basically take part of those inputs, discord part select or they can also say bit select. I can take individual bits of those values that assign them to construct this.

For example, I am taking one of the 16 bits of A, similarly B, C, D, one of the select bits here. And I am also assigning output to one of the output bits here, only out 0 and I replicate it. So this is how we can build, we are building a mux 4 way 16 from a normal mux. Now, the next thing is, so here what we have done, we have taken vectors and basically assigned them, we have taken vectors as inputs and basically taken bits from each of these vectors. So A 0 is one bit of A 16, B0, similarly B0, sel 0, etc.

(Refer Slide Time: 11:54)




```
// File name: projects/01/Mux8Way16.hdl

/**
 * 8-way 16-bit multiplexor:
 * out = a if sel == 000
 *       b if sel == 001
 *       etc.
 *       h if sel == 111
 */

CHIP Mux8Way16 {
    IN a[16], b[16], c[16], d[16],
       e[16], f[16], g[16], h[16],
       sel[3];
    OUT out[16];

    PARTS:
    // Put your code here:
   Mux4Way16(a=a,b=b,c=c,d=d,sel=sel[0..1],out=out1);
   Mux4Way16(a=e,b=f,c=g,d=h,sel=sel[0..1],out=out2);
   Mux16(a=out1,b=out2,sel=sel[2],out=out);

}
```

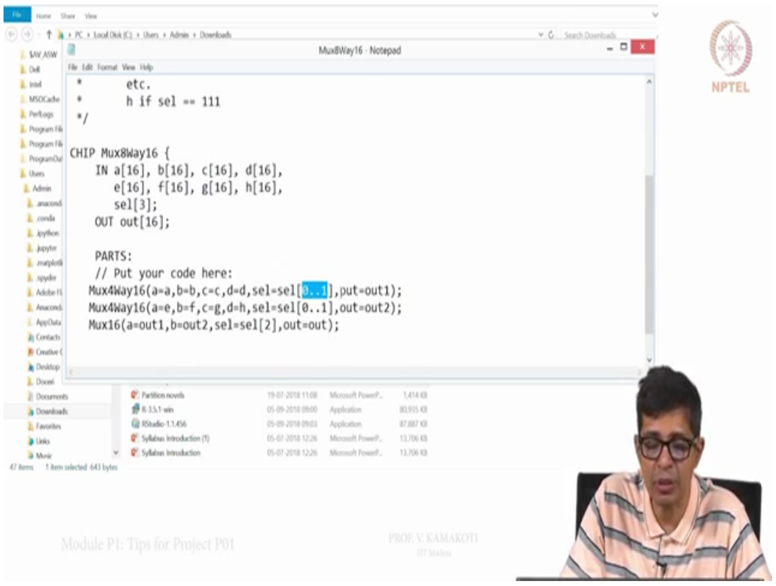
So we have taken one-one bit and we are able to separately assign these bits to some input and outputs. So this is one thing that will help you quickly understand and do the project. The 2nd day that we would like to show here is the 8 way 16-bit mux. 8 way 16-bit mux, so can be easily done here. So, what we do is how do we construct an 8 way mux from a 4 way mux, right, let us say this. So this is a 4 way mux, so how do you construct an 8 way?

The 8 way can be basically constructed as one 4 way, another 4 way, so I will have A, B, C, D, E, F, G, H, and I have sel 0, sel 1, let me call it S1, S0. Then we take this, so each will give me one output and I gave it to 2 way mux and this will be S2. So you have 3 select bits, it is a two-way, 4 way with you have to select bits, 8 way means we will have 3 select bits, S0, S1 and S2 and this will be out. So, if I get 100, 100 is 4, E will go the, 000, A will go there, so this is how. So, I can take two 4 way mux and get this and one to a must to get the 8 remux done.

So that is precisely what we are doing. So, if you look at this, so if you look at, if, if you look at this, right, for the 1st, I am using 1 mux 4 way in which A, B, C, D are taken and then taking the 1st 2 bits of select, select 0 to 1, this is called part select. Select is a 3 bit but I want to use a select 1 and select 0 here. So this mux, what I am doing, this mux for this thing, I am basically using this mux. And for this mux, using only select one and select 0.

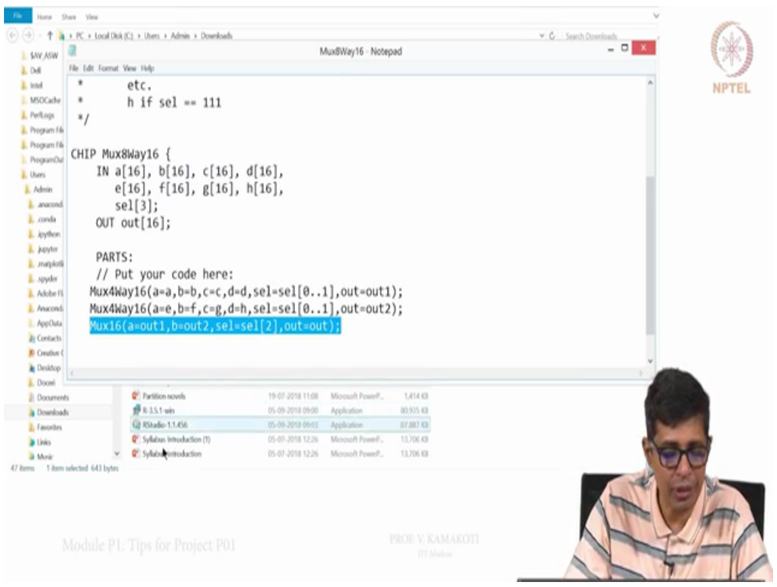
So I need select 0 to 1. Similarly for this mux, again I am taking select 0 to 1 and E, F, G, H as input. And the next mux text the out of the 1st and out of the 2nd and takes only the select 2 bits. So, now this is not just mux 4 way but this is 16-bit mux, 16-bit mux 4 way. So A 16, just put A equal to A, please note that in mux 4 way 16, again I have a 16-bit variable, so all the 16 bits automatically get assign, right. And we are also taking a part of that select, select is 3 bits as you see here, but I am only taking part of that select, so I can also take 2 bits like that, right.

(Refer Slide Time: 15:20)



MuxBWay16 - Notepad

```
 *       etc.
 *       h if sel == 111
 */

CHIP Mux8Way16 {
    IN a[16], b[16], c[16], d[16],
       e[16], f[16], g[16], h[16],
       sel[3];
    OUT out[16];

    PARTS:
    // Put your code here:
   Mux4Way16(a=a,b=b,c=c,d=d,sel=sel[0..1],put=out1);
   Mux4Way16(a=e,b=f,c=g,d=h,sel=sel[0..1],out=out2);
   Mux16(a=out1,b=out2,sel=sel[2],out=out);
```


```
 *       etc.
 *       h if sel == 111
 */

CHIP Mux8Way16 {
    IN a[16], b[16], c[16], d[16],
       e[16], f[16], g[16], h[16],
       sel[3];
    OUT out[16];

    PARTS:
    // Put your code here:
   Mux4Way16(a=a,b=b,c=c,d=d,sel=sel[0..1],out=out1);
   Mux4Way16(a=e,b=f,c=g,d=h,sel=sel[0..1],out=out2);
   Mux16(a=out1,b=out2,sel=sel[2],out=out);
```

So this is another very important thing that we need to keep in mind, this is called part select. In the previous thing when we saw the mux 4 way 16, please know that we were using only bit select, we were taking only one index here. But when we are looking at the next one, that we are showing here, namely all mux 8 way, please note that we are taking part select, that is 2 of the 3 bits. So this is said that that you can do. So select has assigned those 2 bits.

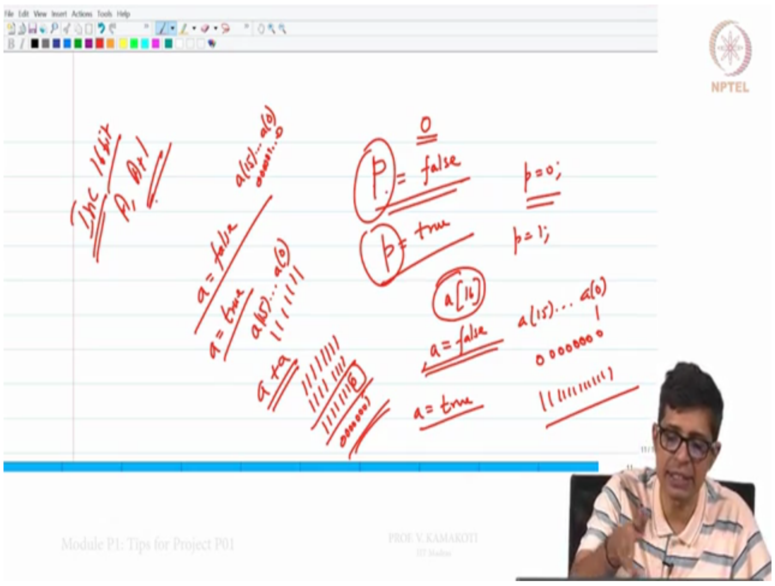
Very interestingly if you also use these out 1 and out 1 are the intermediate wires that we have created. Now this is 16 16 way mux, so this out 1 should be 16-bit, out 2 should also be 16-bit. They automatically become 16-bit because I am assigning it to out and which is defined as 16 bits in mux 4 way 16. In mux 4 way 16, the output variable is 16-bit. So when I assign out 1 to out, automatically out 1 also becomes a 16-bit variable. So, that is very important to note. So suppose I want to generate this 16-bit, put out equal to out 1, since out is 16-bit come out 1 is also assumed to be 16 bits, it becomes 16-bit. Similarly out 2 become 16-bit.

And now I say A equal to out 1, B equal to out 2, all the 16-bits get assigned to A here and all the 16 bits get assigned to B here and use the select, one bit of the select to get the out here. So, the 1st 4 way 16, 4 way 16 will give me 2 outputs, out 1 and out 2, each will be a 16-bit output, that I am going to have put here and depending on the next select I am going to get which of those 16 bits I am interested. So this is how we have basically constructed as you see 3 muxes.

1st mux will give me one of the 1st four A, B, C, D, but 16 bits of them. 2nd mux will give me E, F, G, H, in this case 16-bit because it is a mux 4 way 16-bit. The two 16 bits are put into another mux 16 and which one bit we select which of those 16 bits and is going to take. That point to note here is that the part select, part select, I can select the part of the entire index and also we have seen the select earlier. So these are the things which will quickly enable you to complete the project.

So you will have a lot of queries and all these queries to put back in as the feedback in the discussion forum and we can encourage discussion to actually help you do the project. I hope this lecture, this particular module has basically clarified some of the important inputs on how do we start sending data inside. So this is very important.

(Refer Slide Time: 18:35)



Okay, the next thing, the last thing I want to tell in this module that could be useful to you in the future, also in this particular thing is that how do we generate say 0. Suppose I have a bit, I call it say P, suppose I have a wire P which is a bit, if I say P is equal to false, essentially that meets P is assigned 0. If I say P equal to true, all smalls, the P is assigned 1. Suppose this P is not a bit but it is like what we call like a 16 or something. And I say a equal to false for whatever reason, then a 15 to A0, the entire, all the bits are assigned 0, so this essentially becomes 0000.

If I say a equal to true, then each of these which become one, so I get 111 1111. So if it is a single bit P equal to false, that single bit becomes 0, P equal to true, that single bit becomes 11. But suppose I have thing like a 16 and I say that a equal to false while passing, so this will be while passing some parameters into the design. When I say a equal to false, all the pets become 0, if I say a equal to true, all the bits become 1. So, how will you generate a 0? Just say a equal to false, it will become 0.

How will you generate 1 for example? Suppose I say a equal to true, so let us say a is a 16-bit variable, a equal to false will make a 15 to a 0, all as zeros. Now, a equal to true, if I say a equal to 2, then a 15 to a 0, all will become 1. So when I add a with a, what do I get? Let me say 4 variables, 8 variables, a with a, I get 000… Sorry I get 1111 1110, right. So when I invert this, I get basically. So the way I can generate 1 is take any 16-bit variable, 1st make it true, so all the 16 bits will become 1, add it with itself, then you will land up with a value in which the least significant bit is 0 and all the things are 1.

Now I invert this, so I get the value 1. Right, so this is how we can generate constants like P equal to 0 or P equal to1, etc. So, these are all some things which would be very important for example you will be asked to do an incrementer, right increment it 16-bit incrementer. You are given separate a and you are supposed to write A +1. Now how do you do this incrementer, which would increment it by 1.

How do you do that incrementer? So, again, so things like this can basically come up in a good way. So this is something which is very interesting as far as the project is concerned. With this level of input and if you start working, I am sure you will cross the barrier of project 1 that you need to complete. Thank you.