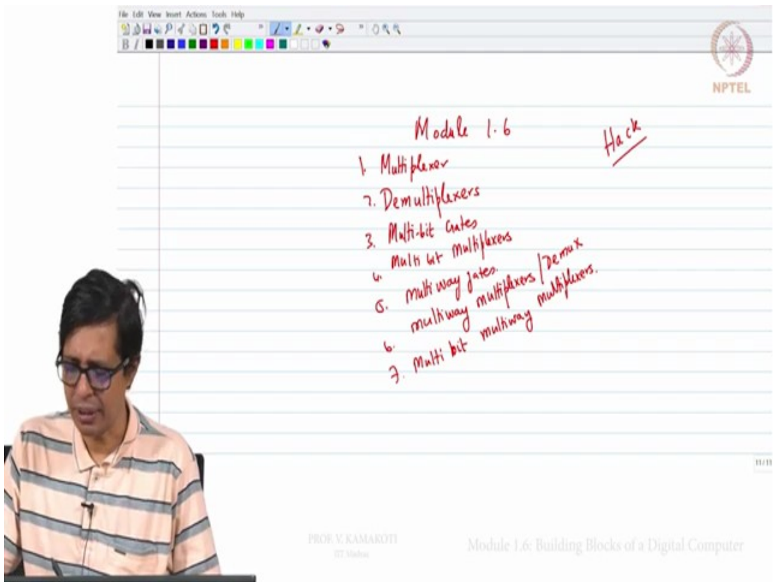
Foundations to Computer System Design

Professor V. Kamakoti
Department of Computer Science and Engineering
Indian Institute of Technology Madras
Module 1.6
Building Blocks of a Digital Computer

(Refer Slide Time: 0:18)



Module 1.6

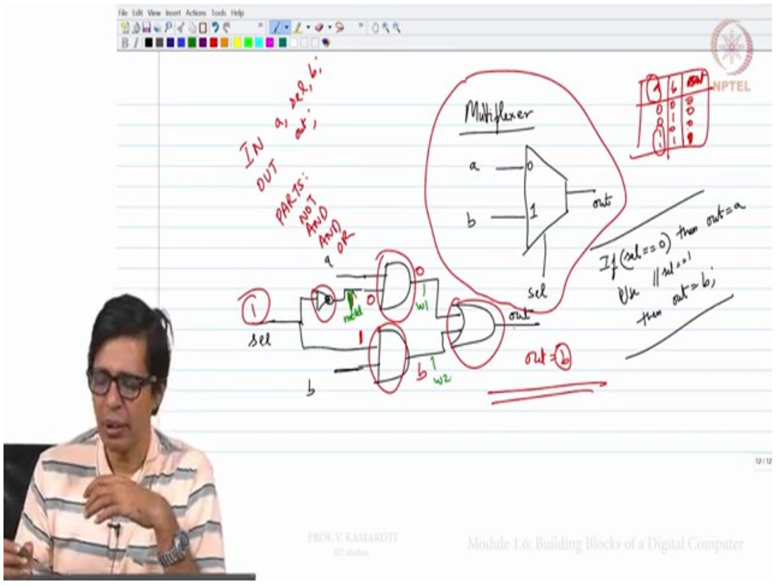
Hack

1. Multiplexer
2. Demultiplexers
3. Multi-bit Gates
4. Multi bit multiplexers
5. multi way gates.
6. multiway multiplexers / Demux
7. Multi bit multiway multiplexers.

So welcome to module 1.6 is the 6th part of module 1 and as we see as part of this course we are going to build a small machine, we are going to build a small hardware we call it hack, the hardware is called hack and that hardware will be simulating as we have seen how to simulate small small gates, we will try and build that hardware and actually simulate that hardware using that hardware simulator that we have been discussing. For that hardware we need many components, we have seen gates but we need a little more complex circuit.

So some of the things that we will be seeing today in this module are multiplexer, demultiplexers, then multi-bit gates, multi-bit multiplexers, multi-way gates, multi-way multiplexers, then a combination of multi-bit multi-way multiplexers. So multiplexers or demultiplexers we will call it demux, so we will see all these 7 different companies as we go to this particular module 1.6. These are basically the building blocks on which you will build the hardware and each requires some small definition. We will just go ahead and do it as a part of this module.

(Refer Slide Time: 2:46)

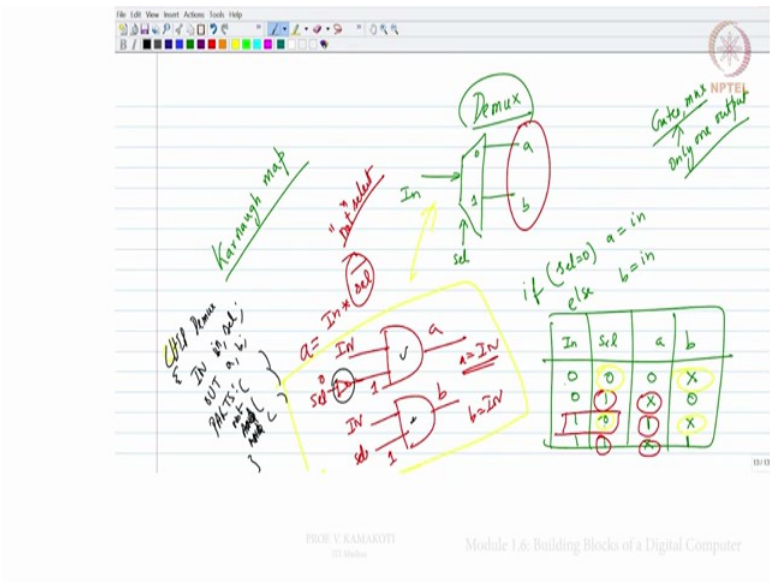


Now what is a multiplexer? It is given by this symbol like for AND gate, OR gate we had so the multiplexer is even like this a, b this is the select and this is the output. Let me call it 01. If select is… this is a single wire and if select is 0 then output is a else select can be 0 or 1, obviously it means select is 1 then out is b. So how do you arrive at the circuit? We arrive at this circuit by just saying this is a, this is b, this is select, this is an inverter and this is out. So this is how we construct a multiplexer very straightforward logic, right? What are we doing?

When select is 0 we have to select a when select is 0 this gate being an NAND gate this input is 0 this is always 0 and this being 1, a and 1 will be a, right? So the truth table for a, b this is NAND gate output 0, 0, 0, 0, 1 is 0 10 is 0 11 is 1, right? Now let us take this a…one of the input is select, right? If this input is permanently 1 meaning these 2 entries 1 and 1 so the output is the other entry B, 0, 1 so if one of the inputs of NAND gate is tied to 1 the output will be the other input. So this is 1, so I get a and this is 0 a, OR 0 is a so when the select is 0 the output is a. Similarly when the select is 1 what will happen? When the selectors one, so this becomes 1, this becomes 0, so this becomes 0, 1 and b is b so out becomes b, so this is how you construct a multiplexer circuit, right?

Now we can easily code this multiplexer, right? So how do we code this? We leave it as a simple exercise, so what are the in? In is basically a, sel and out in that chip in will be a, sel and OUT will be …sorry a, sel, b and OUT will be small out then in PARTS you have to use a NOT gate for this and AND gate for this, another AND gate for this and another OR gate for this and we will have one internal wire, all internal wire I have mark in green, 1 internal wire let me call it as not sel or whatever any name. Let this be W1 this is W2 so 3 internal wires automatically gets reflected in the PARTS, right? So we need to construct a multiplexer using this and this is a very simple exercise.

(Refer Slide Time: 7:28)



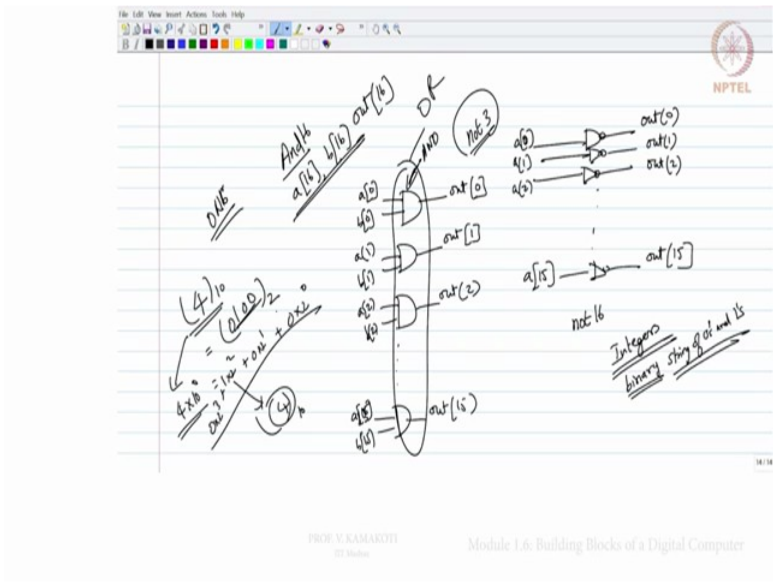
Now let us see the next one it is basically the demultiplexers, what is a demultiplexers? Demultiplexers is opposite of multiplexer, so I get in here if selected is equal to 0 then a is equal to in else b equal to in, okay so this what we need okay so for that this is quite straightforward, so we can write this so this is 2 output circuit. Till now we have been seeing gates and then we saw multiplexer all of them had only one output. Now we are seeing a circuit with 2 outputs as you see here.

So how does that look like, so this is the truth table okay so in 0 when select is 0, a should be in, b can be do not care, we call it do not care as we do not care what is b and when select is 1, b should be 0, a can be anything, select is 0, a should be in, b can be anything, select is 1, b should be in, a can be anything. So this is how this is basically done here, now how do you arrive at the circuit for this, circuit for this is we use for technique all Karnaugh Map technique. I will be explaining Karnaugh Map in some detail in the part of this course but it may not be necessary for us to do Karnaugh Map fully. As a part of this course it is part of digital logic course but we will basically see some orientation to Karnaugh Map as we go through this course.

We will just fair enough to handle the complicities we will get. Now what is this Karnaugh Map, so basically we take do not care what is a, so a I can say is…this is 1, a is nothing but in into select bar or in and select bar, so this is a IN I have the select ((0)(11:09) this is a this combination. In this 1 and select bar, select is 0 that means not of select is 1, select bar means not of select okay now let us see what is a? So when select bar a is needed only when select is 0, when select is 1 we do not care what is the value of a, a is needed when select is 0, so when select is 0 this becomes 1 and a is always equal to in, right? So 1.in 1 and in we have seen already is always in, so when select is 0, a is in, right?

That is what we needed, similarly for b it is very simple so you just put in and select directly, when select is 1, b is in, when select is 0, a is in and when select is 1, b is in. When select is 1 we do not care about in, when select is 1 we do not care about a and when select is 0 we do not care about b as you see here, select is 0 we do not care about b. So we do not really bother what is the b when select is 0, when select is 0 a should be in that is how it has been defined. So this is a very simple circuit that can basically model Demux. So for Demux again we will have chip Demux or whatever and in which in will be in, select out will be a, b and then in PARTS we basically write 1 NOT and then 1 AND another AND sorry 2 AND here 1 AND for this and 1 AND for this and 1 NOT for this, so this is how we define a Demux. So where we will be using Demux and mux we will see later, we may also try to look at this Karnaugh Map as we proceed through the 2nd module.

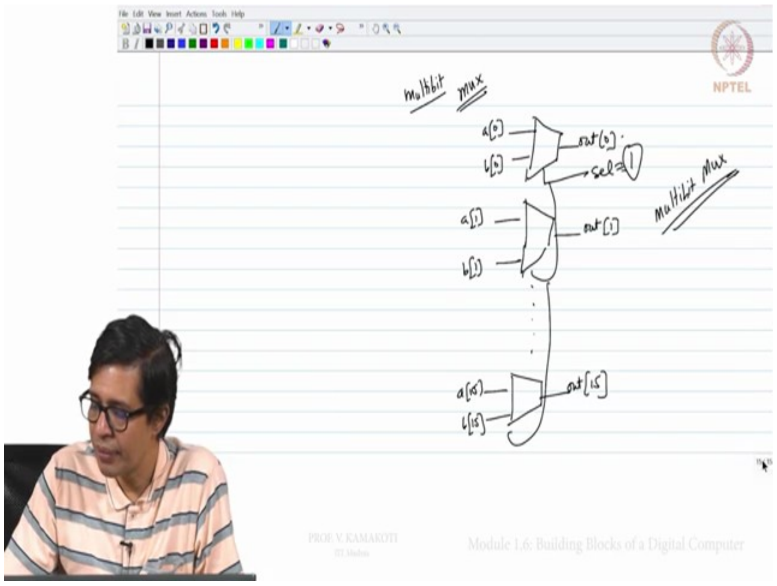
(Refer Slide Time: 14:32)



Now the other small things that we need to basically discuss here as we projected earlier are the multi-bit gates, we have already seen not of 3 but actually we made it as not 16 but we saw not of 3, not of 3 is nothing but 3 gates we give a(1), a(0), a(2) and we saw out(0), out(1), out(2) this can go on till a(15). This is called not16 okay so there are 16 such gates and this is not16 okay. Now we could also have say and 16, or 16 how will and 16 work? We will have 2 sets of input a16 and b16 which are arrays of size 0 to 15, so what will happen we will instantiate 16 AND gates and will also have out16 okay that will be output, so a(0) and b(0) will go here and I will get out(0), a(1) and b(1) will go here we will be out(1), a(2), b(2) will go here and we will get out2 and then we will have a15, b15 going here and it will go as out(15).

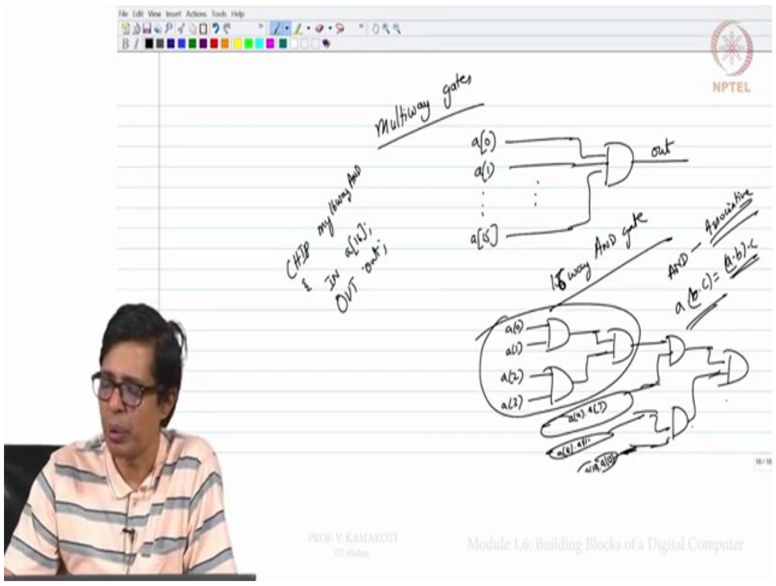
If we can replace all of this by OR currently this is AND if I can replace then it will lay multi-OR gate. Why do we need such gates? Because what will happen is as we are…the computer that you are going to build is going to process a integers, so integers you would have done your basic programming course are represented in binary, it will be a string of 0 and 1, right integers in binary. For example if I say 4 in base 10 this is nothing but 0100 to the base 2, what is 4? 4 into 10 power 0 that is what is 4? In binary this is 0 into 2 power 0 plus 0 into 2 power 1 plus 1 into 2 power 2 plus 0 into 2 power 3, so 1 into 2 power 2 is 4, so this is how… so 0100 is nothing but a multi-bit vector or multi-bit string and if you want to process such multi-bit string we need multi-gates and that is why multi-gates become important for your architecture.

(Refer Slide Time: 17:50)



We will also have multi-bit demux or even multiplexes, so what is 1 mux? So it was taking a b and (())(18:07), right? So we could have a0 b0 out0, a1 b1 out1, a15 b15 out15, right? So this is your multi-bit mux and everything will have the same select signal, so when I make select 0 your out0 will be a0, out1 will be a1, out15 will be a15, if I make select as 1 for example then out0 will become b0, out1 will become b1 and out15 can become b15 so this is what we call as multi-bit mux. So we have seen multi-bit (())(19:15) we have seen multi-bit mux.

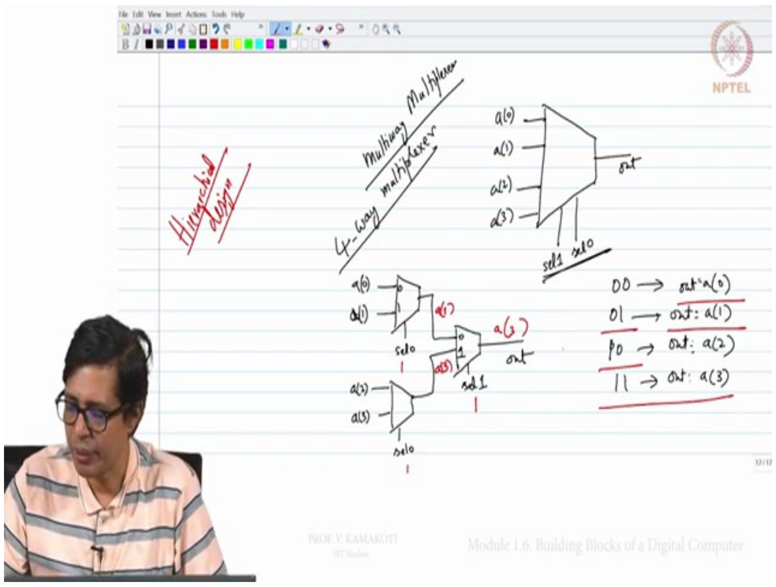
(Refer Slide Time: 19:25)



Now we can see multi-way gates this is nothing but I have one a(0) a(1) to a(15) I want to take the and of this 1 single and of 15 bits or 15 wires. We had seen…so this is a multi-way gates so this is your 16 way gate, 16 way AND gate, okay this is 16 way AND gate. Similarly I can have 16 way OR gate I could have any AND way gate, right. So how do we model this? How do we actually do this? Let me say CHIP my 16 way and whatever so I could have IN as a16 OUT as out, so how do we do this? So basically then we need to…one of the interesting thing is this entire AND operation and being model as a0 a1let me just say till 4, a2 a3 this will be a4 like that I could have…so this is the AND of 4 bits and note that the AND operation is both associative that means a and b and c is equal to a and b and c.

So it does not matter which way I am ending I need not and a[0] then a[1] then a[2] then a[15], I can end any of them in any order and get it, so I just do a[0] a[1] first then a[2] a[3] first and similarly a[4] a[5] a[6] a[7] like that and I can...so this is the way I can AND 4 bits like that I can take 4 such blocks, so I will have 4 such blocks and then I will take these 2 and I will take these 2 again and then finally 1, so let me say this is 1 block so I will have similarly 1 block driving this, similarly another block driving this another block driving this. For this I will have a[4] a[5] a[6] a[7] here a[8], a[9], a[10], a[11] I will have a[12] to a[15] here so I could have multiple blocks driving and then we will get, okay so this is how we can make a multi-way gate, right? So this is also very important thing.

(Refer Slide Time: 23:02)

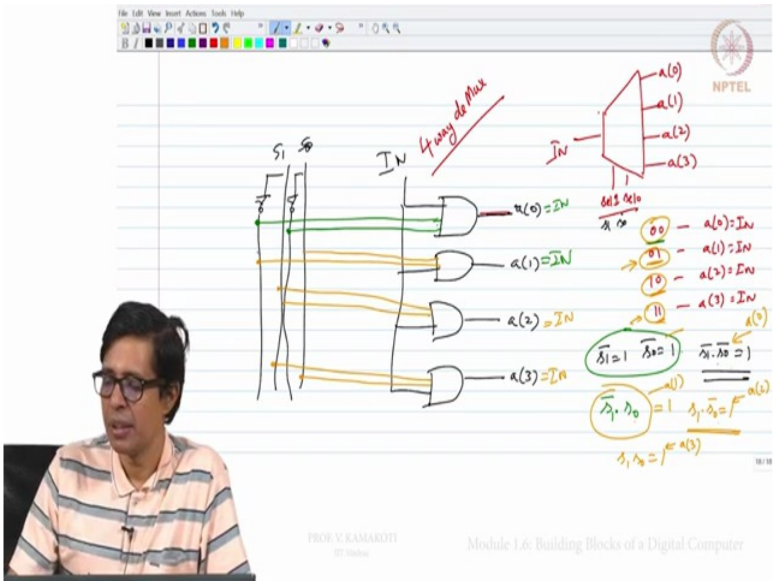


The other thing that we want to look is multi-way multiplexer, how do you do a multi-way multiplexer? So for example I have 4 inputs let me say a(0), a(1), a(2), a(3) and have 2 select lines sel 1 and sel 0 let me call this if this is 00 then the out should be a(0), if this is 01 then the out should be a(1), if it is 10 then the out should be a(2), if it is 11 then the out should be a 2 okay, so this is how we can build their 4 way multiplexer. So how do you get the circuit for a 4 way multiplexer? A 4 way multiplexer can be done using multiple two-way multiplexers, okay. So let us see this…

So how do you construct this you know how to do it 2 way multiplexers already we have seen this, so we will give sel 0 here, we will give sel 0 here and we will give sel 1 here, so I give a0 and a1 for 0 and 1 and a2 and a3 here and this is 0 and 1 and this is out. Now let us say what will happen when I am applying 00 to sel1 and sel0, so this is also 0 basically here a 0 will come here a 2 will come and since this is 0, a 0 comes, right? So when I give 00 I get a 0. Let me say what happens when I give 01?

So suppose I say I am giving 01 that is 0 to sel 0 and 1 to sel 1 sorry 01 is 0 to sel 1 and 1 to sel 0 so I give this, so here a 1 comes and here a 3 comes and since sel 1 is 0 I get a 1, so out is a 1 when I give 01. Suppose I give 10 suppose I give 1 to sel and 0 to this again here a 0 comes and here a 2 comes and since it is 1 I get a 2 here because sel 1 is 1 so a 2 gets in and similarly if I say 11 so if I put 1, so a 1 comes here a 3 comes here and since it is 1 I get 3 so 11 means a 3, so a 4 way multiplexer can be built using 3, 2 way multiplexer. Now you know how to construct a 2 way multiplexer that you use as a chip to build a 4 way multiplexer, so this is what we meant by hierarchical design in the previous module.

(Refer Slide Time: 27:17)



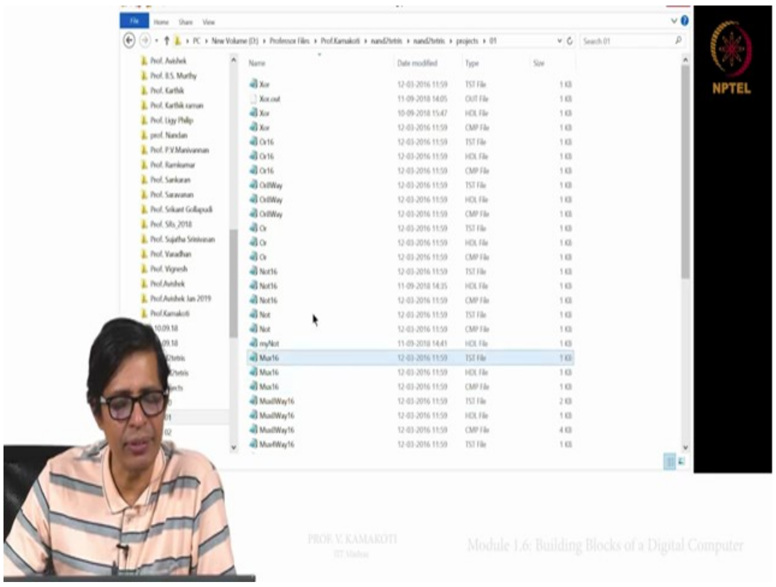
The last one we will be seeing as…let us say I want to make a 4 way demux, what do you mean a 4 way demux? So I have one input then I will put select one and select 0 this is demultiplexers so I have a 0, a 1, a 2, a 3 and I have input when this…I call it select this is 00 then a 0 should be IN 0 1, a 1 should be IN, 10 a 2 should be IN and 11 a 3 should be IN. So how do you construct this circuit? This is quite straightforward, so circuit for that is very simple, so let us just mark these lines a 0, a 1, a 2, a 3 okay and then let us put an AND gate here, it is quite intuitive let us feed IN to all a 0 should get IN let me call it s 1 and s 0 are both 0 that means s 1 bar is 1 and s 0 bar is also 1 and s 1 bar and s 0 bar should be one, so I just say I have s 1, s 0 I also have their s 1 bar and also have their s 0 bar here okay.

Now s 1 bar is 0 bar a very simple way of looking at it, so s 1 bar and s 0 bar both are 1 and IN is 1 so a 0 get IN okay. Now for any other combination of s 1 s 0 except 00 so if any of this s 1 and s 0 are 1 any one of them then this s 1 bar.s 0 bar become 0 because if s 1 is 1 s 1 bar is 00 and anything is 0. Similarly s 0 is 1, s 0 bar is 0, 0.anything is 0, so for any other combination except 00 s 1 bar.s 0 bar will become 0 and essentially a 0 will not get IN it will be 0. Similarly what is a 1, a 1 should get IN when s1 is 0 at means s1bar is 1 and s0 is 1, so I take s1bar from here so let me just take another color, so I take s1bar from here and taking s 0 from here.

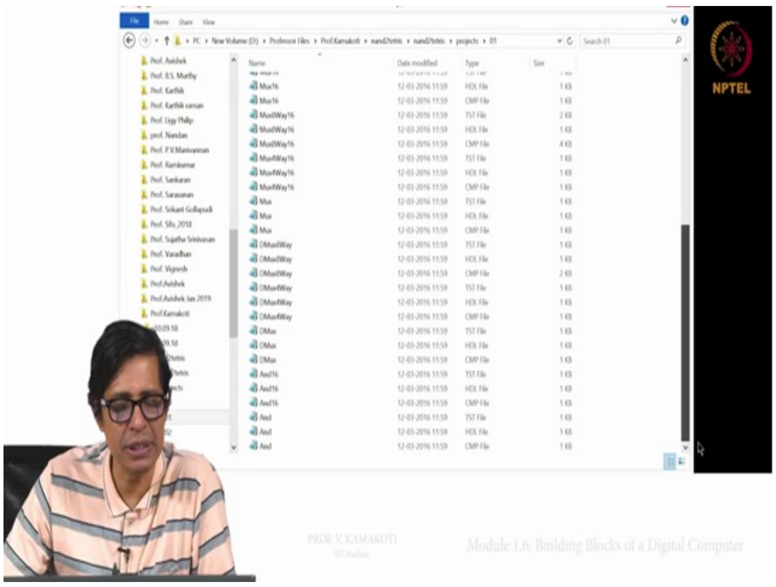
This s1bar.s 0 will be 1 only when s 1 is 0 and s 0 is 1, for every other combination including 00 s 1 bar.s 0 will be 0, right? So only when I get s 1 bar as 0 and s 0 as 1 that is whenever I get 01 then only a 1 will become IN. So now for a 2 it is 10, so s 1 should be 1 and s0 should be 0 that means s0 bar should be 1, so for a 2. I am just taking s1 and I am taking s0bar, so a2 will become IN when s1.s0 bar is 1 that is when s1 is 1 and s0 is 0 and here for this we just take s1 and s0 and your a 3 actually becomes IN when both s1 and s0 are 1.

So this is the condition for a 0 to get IN while this is the condition for a 1 to get IN this is the condition for a 2 to get IN and this is the condition for a 3 to get IN and all these conditions are mutually exclusive that is 2 conditions will not become 1 at the same point of time only exactly 1 will become s1bar.s0bar will become 1 only s1 and s0 are 00 only for this combination it will become 1 for other combinations it will be 0. Similarly s1 bar.s0 will be true only for the 2nd combination and for all the other remaining 3 it will be 0, similarly s 1.s 0 bar for this combination for the remaining it will be 0, s1.s0 for this combination for the remaining it will be 0. Like that we get a 4 way demux.

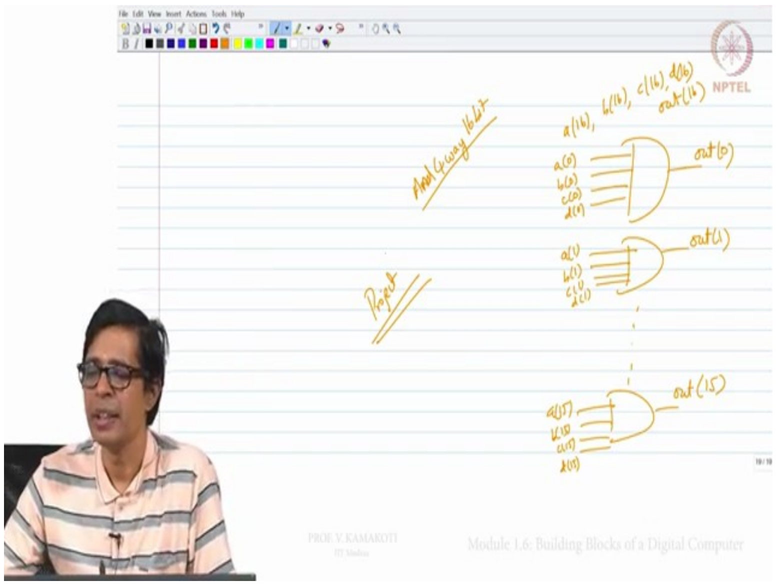
(Refer Slide Time: 33:43)



```
New Volume (D:) › Professor Files › Prof.Kamakoti › nand2tetris › nand2tetris › projects › 01

Name          Date modified      Type      Size
Xor           12-03-2016 11:59   TST File  1 KB
Xor.out       11-09-2018 14:05   OUT File  1 KB
Xor           10-09-2018 15:47   HDL File  1 KB
Xor           12-03-2016 11:59   CMP File  1 KB
Or16          12-03-2016 11:59   TST File  1 KB
Or16          12-03-2016 11:59   HDL File  1 KB
Or16          12-03-2016 11:59   CMP File  1 KB
Or8Way        12-03-2016 11:59   TST File  1 KB
Or8Way        12-03-2016 11:59   HDL File  1 KB
Or8Way        12-03-2016 11:59   CMP File  1 KB
Or            12-03-2016 11:59   TST File  1 KB
Or            12-03-2016 11:59   HDL File  1 KB
Or            12-03-2016 11:59   CMP File  1 KB
Not16         12-03-2016 11:59   TST File  1 KB
Not16         11-09-2018 14:35   HDL File  1 KB
Not16         12-03-2016 11:59   CMP File  1 KB
Not           12-03-2016 11:59   TST File  1 KB
Not           12-03-2016 11:59   CMP File  1 KB
myNot         11-09-2018 14:41   HDL File  1 KB
Mux16         12-03-2016 11:59   TST File  1 KB
Mux16         12-03-2016 11:59   HDL File  1 KB
Mux16         12-03-2016 11:59   CMP File  1 KB
Mux8Way16     12-03-2016 11:59   TST File  2 KB
Mux8Way16     12-03-2016 11:59   HDL File  1 KB
Mux8Way16     12-03-2016 11:59   CMP File  4 KB
Mux4Way16     12-03-2016 11:59   TST File  1 KB
```



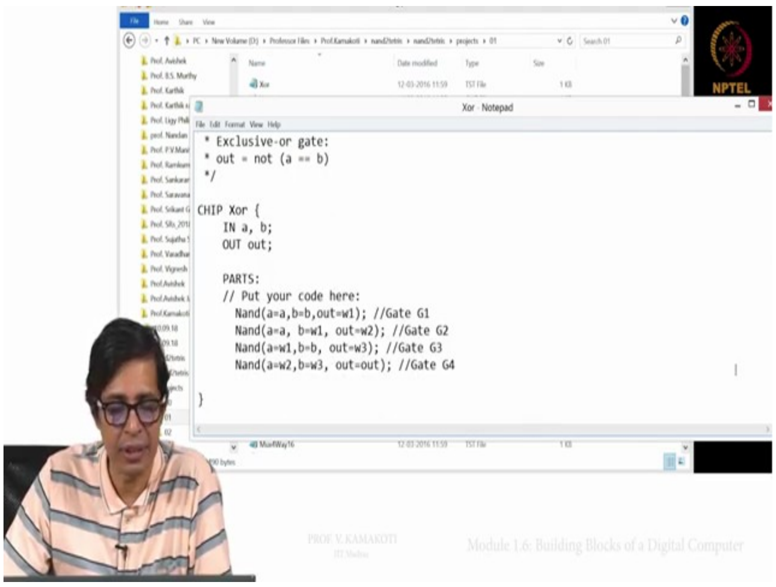
```
New Volume (D:) › Professor Files › Prof.Kamakoti › nand2tetris › nand2tetris › projects › 01

Name          Date modified      Type      Size
Mux16         12-03-2016 11:59   HDL File  1 KB
Mux16         12-03-2016 11:59   CMP File  1 KB
Mux8Way16     12-03-2016 11:59   TST File  2 KB
Mux8Way16     12-03-2016 11:59   HDL File  1 KB
Mux8Way16     12-03-2016 11:59   CMP File  4 KB
Mux4Way16     12-03-2016 11:59   TST File  1 KB
Mux4Way16     12-03-2016 11:59   HDL File  1 KB
Mux4Way16     12-03-2016 11:59   CMP File  1 KB
Mux           12-03-2016 11:59   TST File  1 KB
Mux           12-03-2016 11:59   HDL File  1 KB
Mux           12-03-2016 11:59   CMP File  1 KB
DMux8Way      12-03-2016 11:59   TST File  1 KB
DMux8Way      12-03-2016 11:59   HDL File  1 KB
DMux8Way      12-03-2016 11:59   CMP File  2 KB
DMux4Way      12-03-2016 11:59   TST File  1 KB
DMux4Way      12-03-2016 11:59   HDL File  1 KB
DMux4Way      12-03-2016 11:59   CMP File  1 KB
DMux          12-03-2016 11:59   TST File  1 KB
DMux          12-03-2016 11:59   HDL File  1 KB
DMux          12-03-2016 11:59   CMP File  1 KB
And16         12-03-2016 11:59   TST File  1 KB
And16         12-03-2016 11:59   HDL File  1 KB
And16         12-03-2016 11:59   CMP File  1 KB
And           12-03-2016 11:59   TST File  1 KB
And           12-03-2016 11:59   HDL File  1 KB
And           12-03-2016 11:59   CMP File  1 KB
```



The last thing that we will see before we wind up this module is a 4 way 16 and gate which is 4 way say 16 bit, so that means what, I will have 4 ways so I will have 4 vectors a 16, b 16, c 16 and d 16 and what I need to do is…and I will have some out which is out 16, so this will be out 0, this will be out 1 and this will be out 15 like that we will have, so we will go with a(0), b(0), c(0), d(0). a(1), b(1), c(1), d(1) like that a(15), b(15), c(15), d(15), so this is 4 way 16 bit AND gate. Now you know how to construct a single 4 way gate and that you replicate 16 times to get this okay so these are some of the interesting circuits that you need to do… You need to understand and code.

So what we need to do as a part is 1st I will sign off this 1st part of this course with simple project which is already available in the website you have downloaded with the hardware simulator, so let me just go into that part and explain you, so go to nand2tetris, inside nand2tetris you have a project folder go to 01 where you have several things you have Xor you have Or16 you have Or8 you have just simple Or you have Not16 you have Not then you have Mux16 then Mux8way16, Mux4way16 just simple Mux, Demux 8way, Demux4way just simple Demux, And16 and…for everything you have a test file, so we have the test file HDL file and compare file.

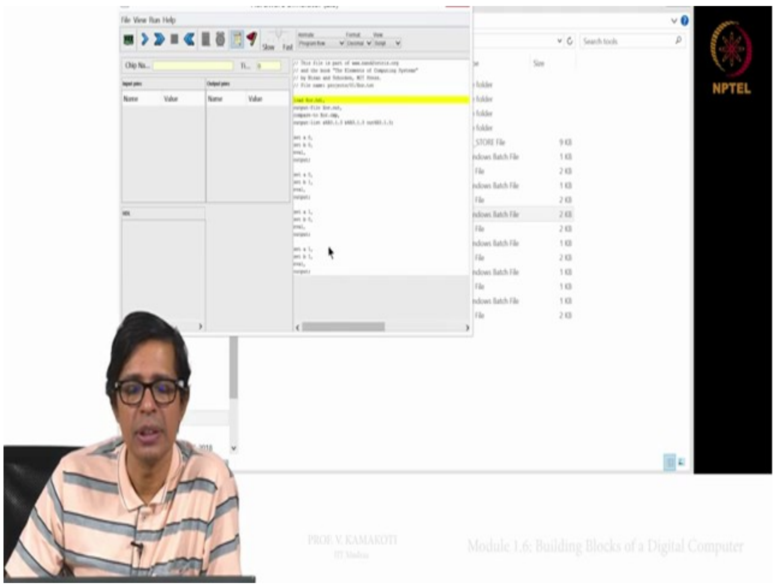
(Refer Slide Time: 36:19)



```
 * Exclusive-or gate:
 * out = not (a == b)
 */

CHIP Xor {
    IN a, b;
    OUT out;

    PARTS:
    // Put your code here:
      Nand(a=a,b=b,out=w1); //Gate G1
      Nand(a=a, b=w1, out=w2); //Gate G2
      Nand(a=w1, b=b, out=w3); //Gate G3
      Nand(a=w2,b=w3, out=out); //Gate G4

}
```

PROF V. KAMAKOTI
IIT Madras

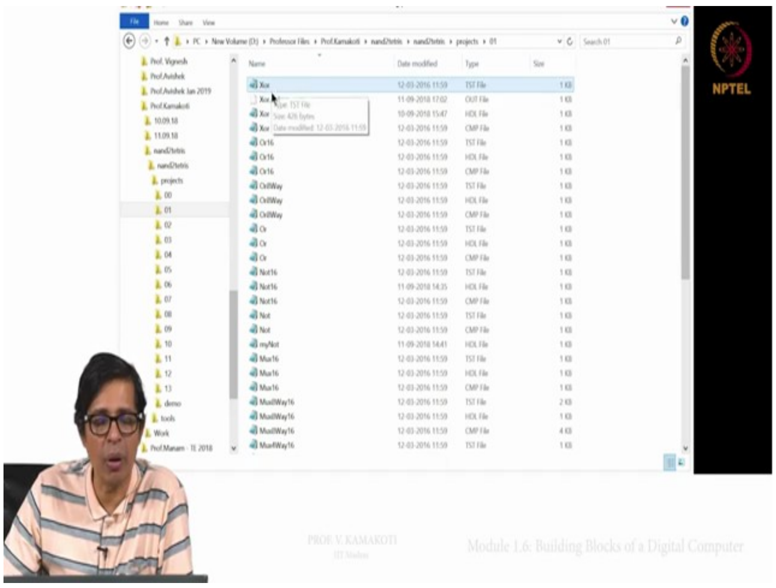
Module 1.6: Building Blocks of a Digital Computer



```
// 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/Xor.tst

load Xor.hdl,
output-file Xor.out,
compare-to Xor.cmp,
output-list a%B3.1.3 b%B3.1.3 out%B3.1.3;

set a 0,
set b 0,
eval,
output;

set a 0,
set b 1,
eval,
output;

set a 1,
set b 0,
eval,
output;

set a 1,
set b 1,
eval,
output;
```

PROF V. KAMAKOTI
IIT Madras

Module 1.6: Building Blocks of a Digital Computer



PC › New Volume (D:) › Professor Files › Prof.Kamakoti › nand2tetris › nand2tetris › projects › 01

Name	Date modified	Type	Size
Xor	12-03-2016 11:59	TST File	1 KB
Xor	11-09-2018 17:02	OUT File	1 KB
Xor	10-09-2018 15:47	HDL File	1 KB
Xor	12-03-2016 11:59	CMP File	1 KB
Or16	12-03-2016 11:59	TST File	1 KB
Or16	12-03-2016 11:59	HDL File	1 KB
Or16	12-03-2016 11:59	CMP File	1 KB
Or8Way	12-03-2016 11:59	TST File	1 KB
Or8Way	12-03-2016 11:59	HDL File	1 KB
Or8Way	12-03-2016 11:59	CMP File	1 KB
Or	12-03-2016 11:59	TST File	1 KB
Or	12-03-2016 11:59	HDL File	1 KB
Or	12-03-2016 11:59	CMP File	1 KB
Not16	12-03-2016 11:59	TST File	1 KB
Not16	11-09-2018 14:35	HDL File	1 KB
Not16	12-03-2016 11:59	CMP File	1 KB
Not	12-03-2016 11:59	TST File	1 KB
Not	12-03-2016 11:59	CMP File	1 KB
myNot	11-09-2018 14:41	HDL File	1 KB
Mux16	12-03-2016 11:59	TST File	1 KB
Mux16	12-03-2016 11:59	HDL File	1 KB
Mux16	12-03-2016 11:59	CMP File	1 KB
Mux8Way16	12-03-2016 11:59	TST File	2 KB
Mux8Way16	12-03-2016 11:59	HDL File	1 KB
Mux8Way16	12-03-2016 11:59	CMP File	4 KB
Mux4Way16	12-03-2016 11:59	TST File	1 KB

Type: TST File
Size: 426 bytes
Date modified: 12-03-2016 11:59

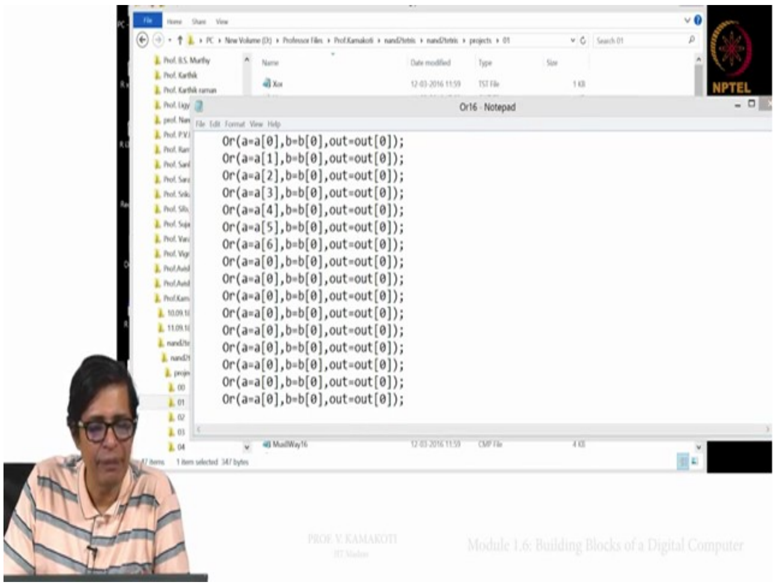
PROF V. KAMAKOTI
IIT Madras

Module 1.6: Building Blocks of a Digital Computer

So if you go to the HDL file in this we just wrote the code but you will not see this code in your thing you have to write the code here and what you need to do you have to write the code and after writing the code save it here and there is already a compare file for this, there is a test file here okay so what you need to do is you open the hardware simulator and after so you have to take this HDL file and fill this part here. Once you fill this part then what you can do is that go to the hardware simulator which is inside the tools, if you are a Linux user use hardware simulator.sh, a Windows user use hardware simulate that opens.

Now you go in and now you can go back and see…go to your projects 01 and we need not load the chip actually we can load the script directly so we can go back to this go to the projects, go to 01 or example you have done XOR so take the XOR.txt and you load the script automatically that will take care of HDL out as I explained in the previous part and then you run it. When you run and it should say end of script comparison and that is successfully as you see here in this part of this, right? So this not only sees this there is a compared script as I told this would be compared your actual output will be compared with that and shown that both are matching okay so like this you need to do for all the project 1 which is very… I am sure it is going to be interesting and it is also going to be simple, right? For all this Xor, Or16, et cetera.

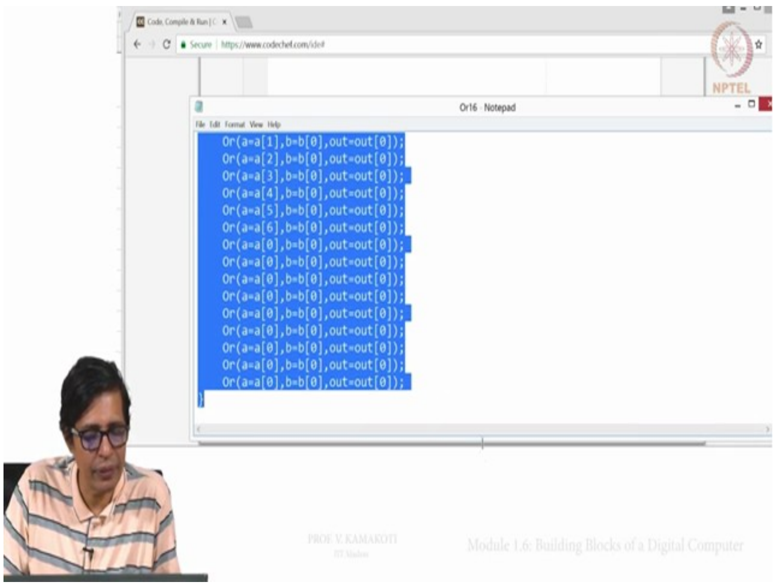
(Refer Slide Time: 38:49)



```
    Or(a=a[0],b=b[0],out=out[0]);
    Or(a=a[1],b=b[0],out=out[0]);
    Or(a=a[2],b=b[0],out=out[0]);
    Or(a=a[3],b=b[0],out=out[0]);
    Or(a=a[4],b=b[0],out=out[0]);
    Or(a=a[5],b=b[0],out=out[0]);
    Or(a=a[6],b=b[0],out=out[0]);
    Or(a=a[0],b=b[0],out=out[0]);
    Or(a=a[0],b=b[0],out=out[0]);
    Or(a=a[0],b=b[0],out=out[0]);
    Or(a=a[0],b=b[0],out=out[0]);
    Or(a=a[0],b=b[0],out=out[0]);
    Or(a=a[0],b=b[0],out=out[0]);
    Or(a=a[0],b=b[0],out=out[0]);
    Or(a=a[0],b=b[0],out=out[0]);
    Or(a=a[0],b=b[0],out=out[0]);
```

So let us go and see one more thing before we wind up let us say I want to do Hindi projects in 01 let me say I want to do Or16 this is script file. Let us say I am going to do Or16 here so how do we do here? So now I can say that I will put Or a equal to a0, b equal to b0, out equal to out 0, right? So this you have to keep repeating again and again for all the…so we just copy this and then paste it so this…Now we have got 2 versions now you can take this 2, copy and paste now we have 4, now copy this 4 and then paste this is very easy way of coding this. Now we have got 8 now copy this and then paste so now you have 16, now just go and make 1, 2, 3 like that you can replace everything 4, 5, 6 but this is really taxing, right? This is boring, so what can we do for doing such type of things?

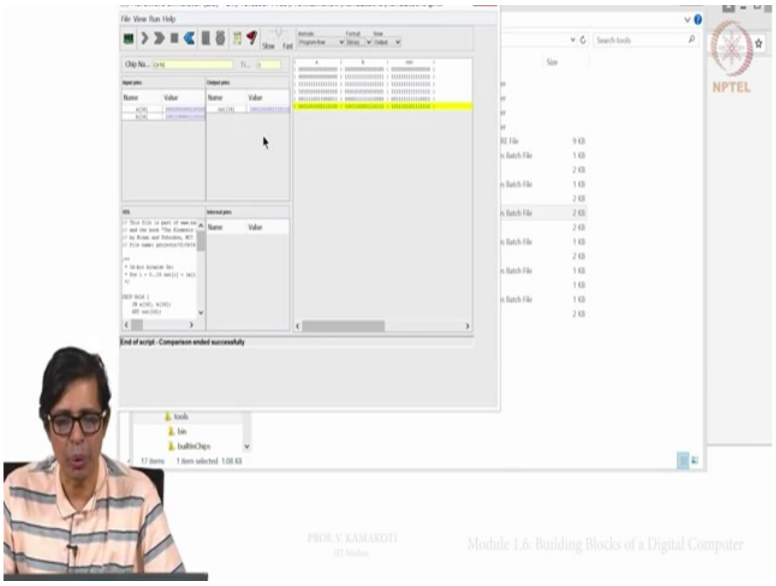
(Refer Slide Time: 40:43)



```
    Or(a=a[1],b=b[0],out=out[0]);
    Or(a=a[2],b=b[0],out=out[0]);
    Or(a=a[3],b=b[0],out=out[0]);
    Or(a=a[4],b=b[0],out=out[0]);
    Or(a=a[5],b=b[0],out=out[0]);
    Or(a=a[6],b=b[0],out=out[0]);
    Or(a=a[0],b=b[0],out=out[0]);
    Or(a=a[0],b=b[0],out=out[0]);
    Or(a=a[0],b=b[0],out=out[0]);
    Or(a=a[0],b=b[0],out=out[0]);
    Or(a=a[0],b=b[0],out=out[0]);
    Or(a=a[0],b=b[0],out=out[0]);
    Or(a=a[0],b=b[0],out=out[0]);
    Or(a=a[0],b=b[0],out=out[0]);
    Or(a=a[0],b=b[0],out=out[0]);
}
```

Actually can go to some website, so let us say online C programming practice or whatever so you can go to codechef and write a very simple program as you see here, so I equal to 0 to 16 just ask it to print this is what you need? a is a 0, b is b 0, out is out 0, a is a 1, b is b 1, out is out 1, so just go here and say run type this code and run and already you see the outputs here, so just go here just copy this come back here paste it here…so this is not going that friendly so what we need to do is just putting enter here, done. That is it so we have done that very nicely so we can now go and save this.

(Refer Slide Time: 43:05)



End of script - Comparison ended successfully

Now we can go back to your hardware simulator and you can load your Or16, actually we can load your Or16. Instead of doing that we can also load your script directly Or16 script, now see this giving 000 as input it is getting 000 2 inputs then 01,11, 10, 10 so it is giving 6 different 16 inputs here, now we can say that let us go and run this comparison and that is successfully done. Now we can see or output so 16 a and 16 b all are 0 you get 0, a is 0, b is 1 then you get 1, a are 1 and b are 1 you get 1, here 10,10,10 and 01, 01, 01 again and you get 111 some random values and you can check it is been compared correctly.

So this is a very nice way by which you write a C program to generate hardware description you actually wrote a C program to basically generate this Or16 so you can basically do such type of things and also note some random values are coming here, do not worry this format of the output if you make it as binary then it will give you exactly what is it, that also please note. So now you are expected to do this project 01 as a part of this module 1, go to that project 01 and completely do all the exercises there and run the script and basically see if things are working and then the model 2 will become very easy for you. Thank you very much!