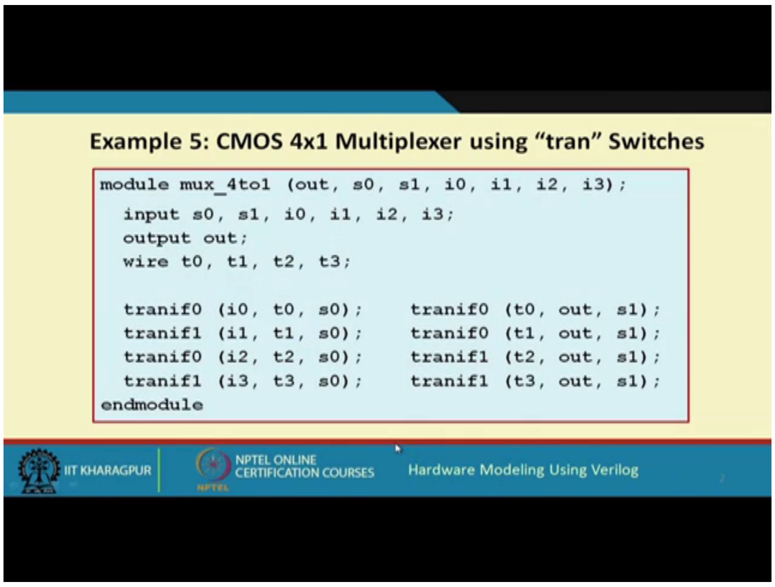
Hardware Modeling using Verilog

Prof. Indranil Sengupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 36
Switch Level Modeling (Part 2)

So, in this lecture we continue our discussion on switch level modelling in verilog. You recall in our last lecture we talked about the NMOS, PMOS and the CMOS switches, we took some examples, some simple gate implementations. And we also saw how a bidirectional switch works, but we did not see any examples of that. So, today we shall be starting with giving some examples of such bidirectional switches and some other things. So, the topic of this lecture is switch level modelling the second part ok.

(Refer Slide Time: 01:03)



Example 5: CMOS 4x1 Multiplexer using "tran" Switches

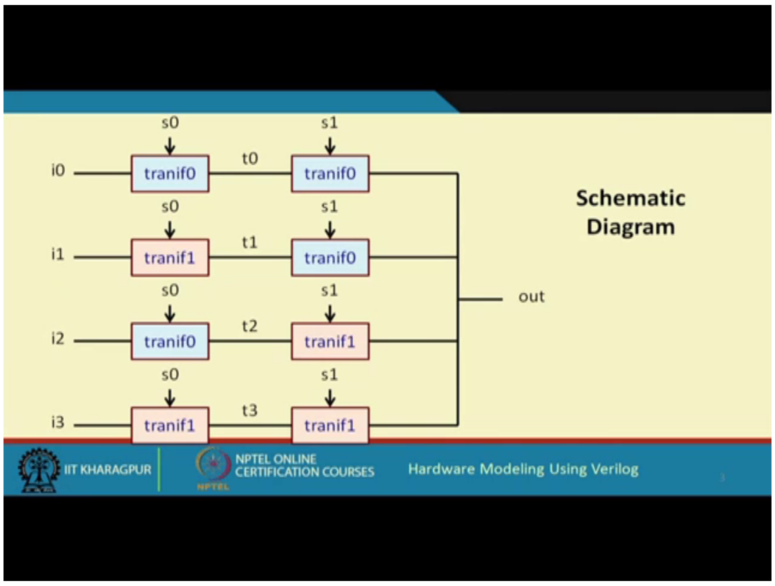
```
module mux_4to1 (out, s0, s1, i0, i1, i2, i3);
  input s0, s1, i0, i1, i2, i3;
  output out;
  wire t0, t1, t2, t3;

  tranif0 (i0, t0, s0);     tranif0 (t0, out, s1);
  tranif1 (i1, t1, s0);     tranif0 (t1, out, s1);
  tranif0 (i2, t2, s0);     tranif1 (t2, out, s1);
  tranif1 (i3, t3, s0);     tranif1 (t3, out, s1);
endmodule
```

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Hardware Modeling Using Verilog

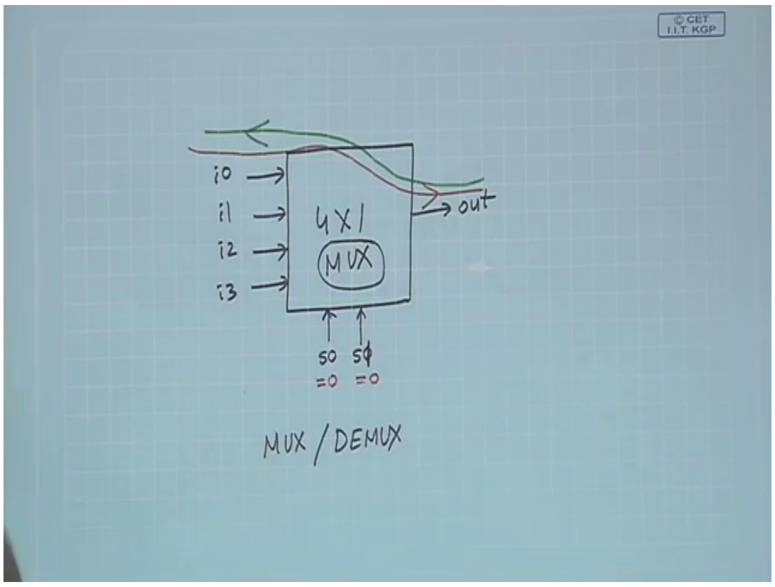
So, here what we try to do is we are trying to implementing a 4 by 1 multiplexer using by directional switches, but let us look into this verilog code later.

(Refer Slide Time: 01:23)



First let us see the schematic how we are trying to design. Here we are trying to implement a 4 line to one line multiplexer ok.

(Refer Slide Time: 01:39)



You just try to understand the functionality in a 4 to 1 multiplexer. So, you have the 4 inputs i 0 i 1 i 2 and i 3. So, we have the output let us call it out and the 2 select lines. So, s 0 and s 1 s 0 and s 1 depending on the select line one of the inputs will be connected to the output.

Now, here because we are using bidirectional switches, we are actually not just implementing marks we are implementing multiplexer come de multiplexer. What is that mean? This means suppose I have applied s 0 equal to 0 and s 1 equal to 0 both 0; that means, i 0 is selected which means from input i 0 current will flow to out. Not only that because the switches are bidirectional from the output node, current will also flow into i 0.

Similarly, if I apply some other control signal this current will get diverted to that particular input and this will be bidirectional current can flow in any direction right. So, although I have called it a multiplexer it is actually a multiplexer cum demultiplexer. Multiplex means many to one de multiplex means one to many both ways ok.

Now, let us see how it works. Here we have used a combination of tran if 0 and tran is 1 switches. Now you recall what a tran if 0 switches. A tran if 0 switch means if the control signal is 0 if 0 then it will be conducting. And a tran is 1 switch means if the control signal is one then it will be conducting ok.

Now, let us see. The 4 inputs are i 0 i 1 i 2 and i 3 and this is the out. Now when I have to select i 0 you see this s 0 I have connected to all the switches in the first level. And s 1 I have connected to all the switches in the second level. So, if I want to select i 0. So, s 0 s 1 will be both 0 and 0. You see the first row of the structure. I have connected to tran if 0 switches if s 0 is 0 this will be on if s 1 is 0 this will be on. So, when both s 0 s 1 are 0 this path will be on and i 0 and out will be connected.

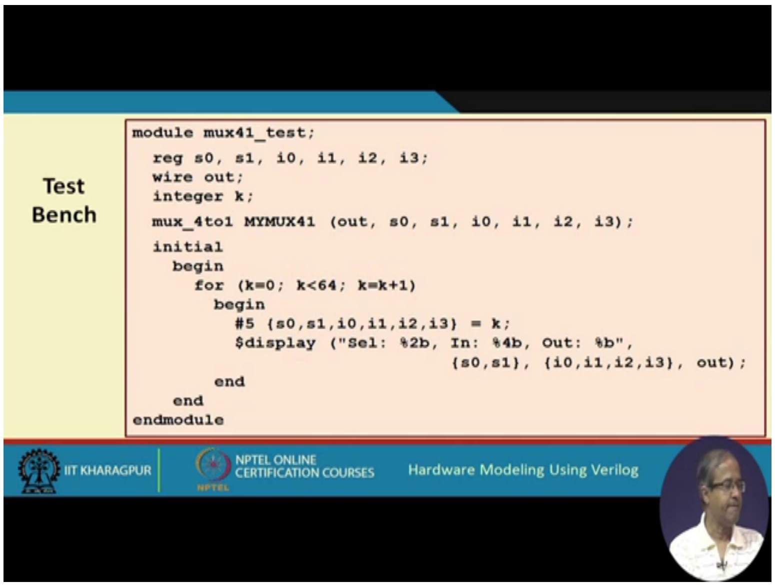
Now, suppose s 1 is 0 and s 0 is 1 then i 1 should be selected. That is why we have use a tran is 1 switch here. If s 0 is one and s 1 is 0 then this path will be selected. Similarly if s 1 is 1 and s 0 is 0 then i 2 is selected. And if both of them are 1 then the last one is selected.

So, depending on the value of s 0 and s 1, exactly one of the row is selected and that input is getting connected to the output right. Now this has been coded directly in verilog here. You see there are 8 tran switches, 4 and 4 8. This is a multiplexer out s 0 s 1 are the select lines and i 0 i 1 i 2 i 3 are the inputs.

So, the intermediate line these one t 0 t 1 t 2 t 3 they are declared as wires. So, we have just simply coded you see tran if 0, first row has 2 tran if 0 switches. So, i 0 to t 0 and t 0 to out. You see i 0 to t 0 and t 0 to out. Second one has a tran is 1 and tran if 0, this is tran is 1 tran if 0. This is i 1 to t 1 and t 1 to out. Similarly i 2 to t 2, t 2 out I t t 3 t 3 out. So, this exactly implements this net list.

Now, here you see in this circuit there will be 6 inputs right, 4 inputs and 2 select lines. So, in 6 inputs there can be 64 total combinations.

(Refer Slide Time: 06:44)



Test Bench

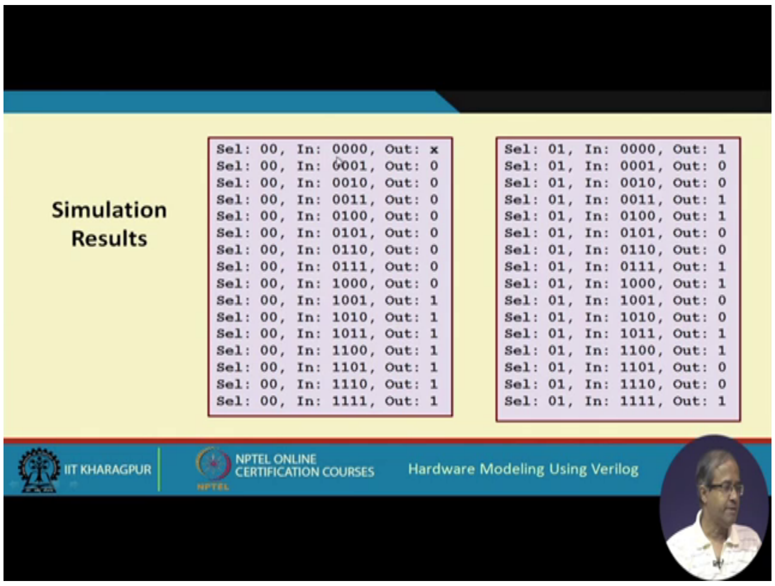
```
module mux41_test;
  reg s0, s1, i0, i1, i2, i3;
  wire out;
  integer k;
  mux_4to1 MYMUX41 (out, s0, s1, i0, i1, i2, i3);
  initial
    begin
      for (k=0; k<64; k=k+1)
        begin
          #5 {s0,s1,i0,i1,i2,i3} = k;
          $display ("Sel: %2b, In: %4b, Out: %b",
                           {s0,s1}, {i0,i1,i2,i3}, out);
        end
    end
endmodule
```

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Hardware Modeling Using Verilog

So, here we have simulated this also using a test bench, where we have instantiated this multiplexer. All this 6 inputs have declared as reg, and in a similar style in a for loop we are generating all this 64 patterns 0 up to 63. And these 6 variables s 0 s 1 i 0 i 1 i 2 i 3 we are assigning the value of k to that.

So, it will be converted to binary the last 6 bits of k will be assigned here. And you are displaying the value of s 0 and s 1 in 2 bit binary cell 4 bit inputs in and the output. So, this loop will be running 64 times right.

(Refer Slide Time: 07:34)



Simulation Results

```
Sel: 00, In: 0000, Out: x
Sel: 00, In: 0001, Out: 0
Sel: 00, In: 0010, Out: 0
Sel: 00, In: 0011, Out: 0
Sel: 00, In: 0100, Out: 0
Sel: 00, In: 0101, Out: 0
Sel: 00, In: 0110, Out: 0
Sel: 00, In: 0111, Out: 0
Sel: 00, In: 1000, Out: 0
Sel: 00, In: 1001, Out: 1
Sel: 00, In: 1010, Out: 1
Sel: 00, In: 1011, Out: 1
Sel: 00, In: 1100, Out: 1
Sel: 00, In: 1101, Out: 1
Sel: 00, In: 1110, Out: 1
Sel: 00, In: 1111, Out: 1
```

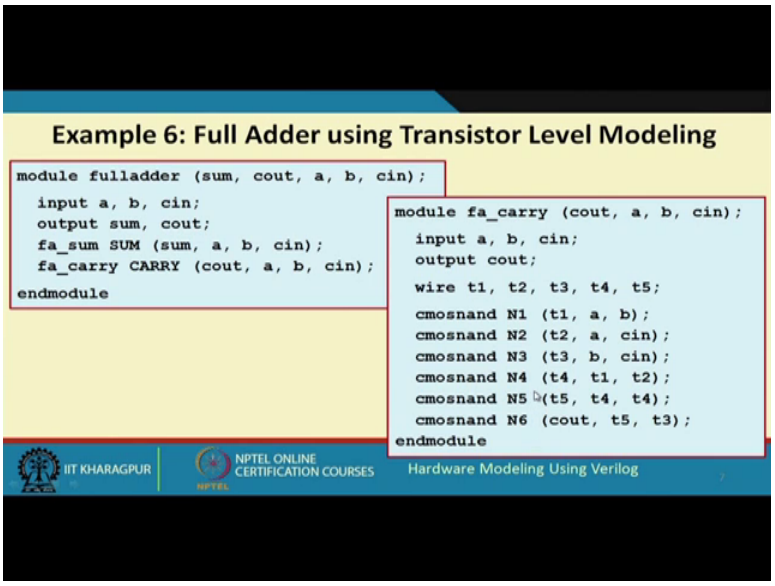
```
Sel: 01, In: 0000, Out: 1
Sel: 01, In: 0001, Out: 0
Sel: 01, In: 0010, Out: 0
Sel: 01, In: 0011, Out: 1
Sel: 01, In: 0100, Out: 1
Sel: 01, In: 0101, Out: 0
Sel: 01, In: 0110, Out: 0
Sel: 01, In: 0111, Out: 1
Sel: 01, In: 1000, Out: 1
Sel: 01, In: 1001, Out: 0
Sel: 01, In: 1010, Out: 0
Sel: 01, In: 1011, Out: 1
Sel: 01, In: 1100, Out: 1
Sel: 01, In: 1101, Out: 0
Sel: 01, In: 1110, Out: 0
Sel: 01, In: 1111, Out: 1
```

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Hardware Modeling Using Verilog

So, there will be 64 lines of outputs. So, I am showing it into slides this is first 16 lines. So, in sel is 0 0 n is 0 0 up to this you see the first line in 0 is getting selected this is in 0.

Now, you can ask why the first one is coming as x. You see the reason is as follows see I means you are displaying it here right and see sel is 0 in is 0. So, sel is 0 n is 0. So, you are giving a delay of 5 and then you are assigning the value of k, but when you are displaying it there is no delay there. So, this will be displaying the previous value that is why before the assignment is done you are getting displayed. Similarly when select is 0 1 you see the second input is getting selected right; similarly 1 0, similarly 1 1 right fine.

(Refer Slide Time: 08:44)


Example 6: Full Adder using Transistor Level Modeling

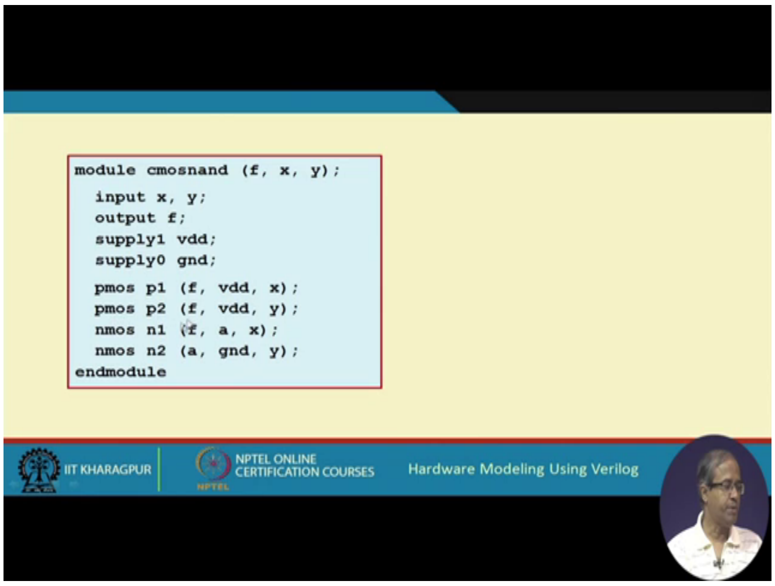
```
module fulladder (sum, cout, a, b, cin);
  input a, b, cin;
  output sum, cout;
  fa_sum SUM (sum, a, b, cin);
  fa_carry CARRY (cout, a, b, cin);
endmodule
```

```
module fa_carry (cout, a, b, cin);
  input a, b, cin;
  output cout;
  wire t1, t2, t3, t4, t5;
  cmosnand N1 (t1, a, b);
  cmosnand N2 (t2, a, cin);
  cmosnand N3 (t3, b, cin);
  cmosnand N4 (t4, t1, t2);
  cmosnand N5 (t5, t4, t4);
  cmosnand N6 (cout, t5, t3);
endmodule
```

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Hardware Modeling Using Verilog

Now, let us see slightly more complex example of a transistor level modelling. Now here we are trying to implement a full adder using transistor level modelling. So, the top level module says that for the full adder we are using a sum module and a carry module. So, f a sum and f a carry. So, f a carry we are directly implementing using this CMOS nand.

(Refer Slide Time: 09:23)

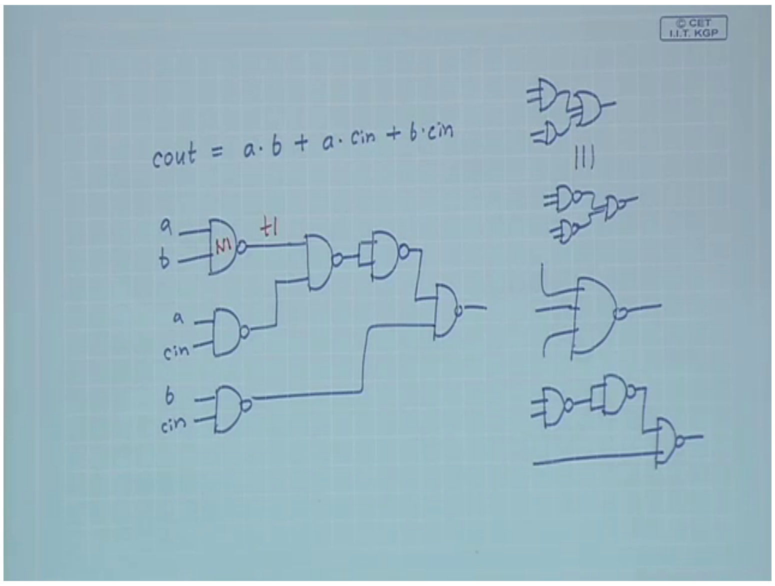

```
module cmosnand (f, x, y);
  input x, y;
  output f;
  supply1 vdd;
  supply0 gnd;
  pmos p1 (f, vdd, x);
  pmos p2 (f, vdd, y);
  nmos n1 (f, a, x);
  nmos n2 (a, gnd, y);
endmodule
```

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Hardware Modeling Using Verilog

You see CMOS nand is something which you have implemented, later we shall come to it CMOS nand you already seen earlier right using 2 PMOS and 2 NMOS transistors can implement a nand gate. So, this CMOS nand is already there we have already implemented this.

So, using that we are using 6 nand gates to implement the carry, how you see for a for the carry function.

(Refer Slide Time: 09:50)



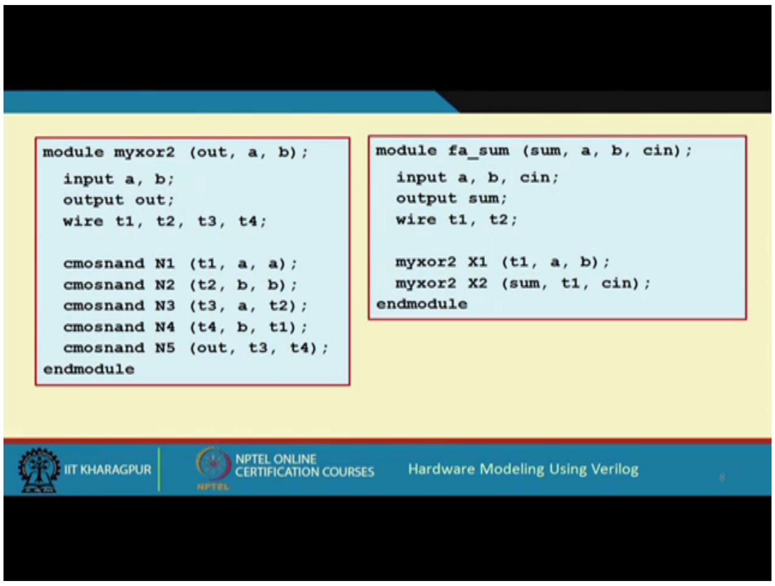
So, how does carry function is implement? A and b or a and c in or b and c in. This is the carry function. So, it is a 2 level and or circuit.

Now, you know any 2 level and or circuit is functionally equivalent to a 2 level nand nand circuit. So, we are using this principle, this we are implementing as a 2 level nand nand. So, in a first level we are connecting a b, we connect a and c in we connect b and c in. Next level there has to be a nand gate there has to be a nand gate that will be taking the 3 inputs, but our CMOS nand gate is only a 2 input nand gate. So, how to implement a 3 input nand gate? 3 input nand gate we implement it like this. We take a 2 input nand gate we do or not by connecting the same input and using another 2 input nand gate.

So, exactly we have done this, we have connected the first 2 to a and gate then we have use used another nand gate for inversion and a third nand gate connected like this. There are total 6 nand gates. So, exactly we have done or created that net list here, this you can verify and t 1 t 2 t 3 t 4 t 5 will be this intermediate line there are 5 intermediate lines you can see one 2 3 4 and 5.

So, I leave it as a exercise for you to verify that whether this net list as I have specified is correct or not and which one is n 1 which one is n 2 and so on this you can see like a for example, let me tell you one of them see this is n 1 t 1 a b. T 1 a b will be this a b is this this is t 1 and this gate is n 1 right. Similarly with others you can identify it right.

(Refer Slide Time: 12:37)



```
module myxor2 (out, a, b);
  input a, b;
  output out;
  wire t1, t2, t3, t4;

  cmosnand N1 (t1, a, a);
  cmosnand N2 (t2, b, b);
  cmosnand N3 (t3, a, t2);
  cmosnand N4 (t4, b, t1);
  cmosnand N5 (out, t3, t4);
endmodule
```

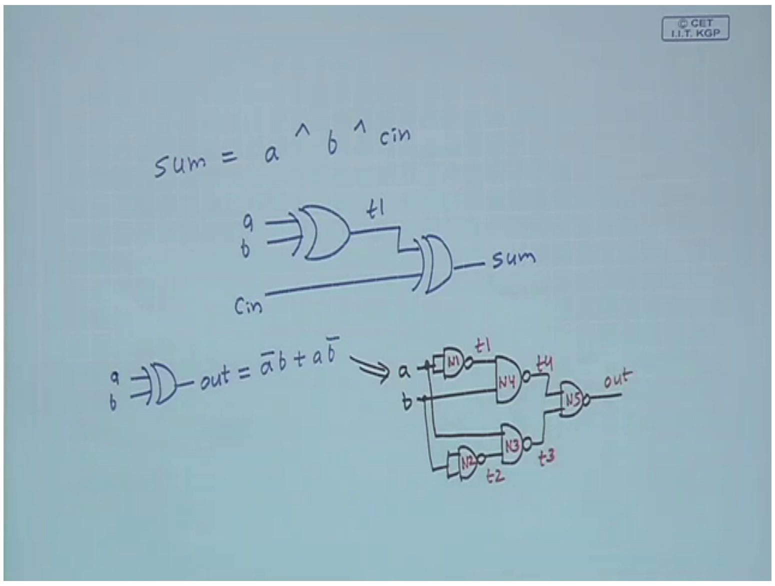
```
module fa_sum (sum, a, b, cin);
  input a, b, cin;
  output sum;
  wire t1, t2;

  myxor2 X1 (t1, a, b);
  myxor2 X2 (sum, t1, cin);
endmodule
```

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Hardware Modeling Using Verilog

Now, after we have implemented carry then we have to implement the sum. Sum at the higher level you implemented using 2 x or gates.

(Refer Slide Time: 12:46)



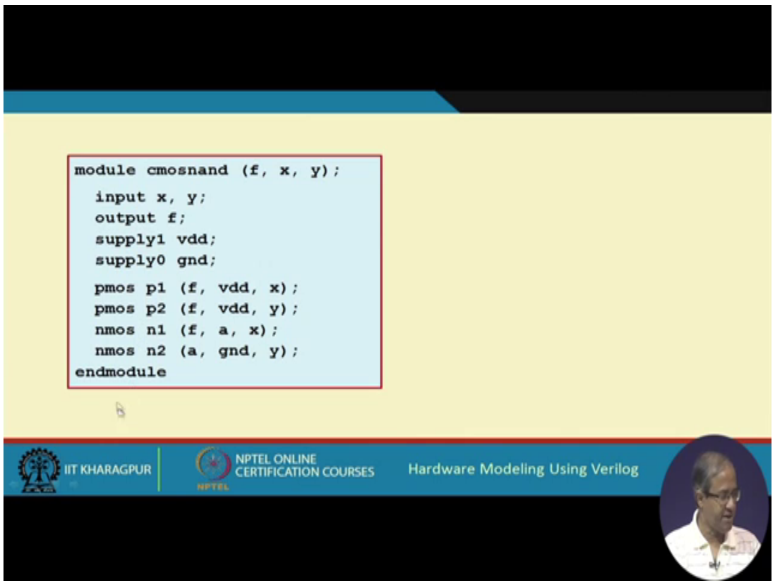
Because sum is what? Sum is nothing but sum is a x or b x or c in. So, you can implement it using 2 x ors say first you can x or let say a and b, you generate let us say a temporary t 1 then use another x or connect c in to it you generate sum right.

So, we have used this kind of a configuration this net list to express f a sum in a structured way. There are 2 x or gates one of them is taking a b, and generating t 1 other is taking t 1 and c in and generating sum. Now these are 2 input x ors well, again 2 input x or I have to implement using CMOS.

Now, let us see. So, how to implement a 2 input x? Or let say a b and out. So, the x or function is given as a bar b or a b bar. So, again using 2 level gates nand nand realisation, you can implement this using this is a bar b. Suppose this is your a you make it a bar with a nand gate make it inverted a bar and b, this is b and a b bar this a you connect directly here a and b bar. So, b you take make a knot of it b bar or. So, or 2 level and or is equal to 2 level nand, nand. This will be your recovery net list 5 nand gates 2 input nand gates. So, here we have exactly done that you see 5 nand gates.

Let us identify the gates here CMOS nand n 1 t 1 a. A a a is this one, this one will be t 1 this is n 1 second n 2 is b b and the output is t 2 this one t 2 b and b this is n 2. And n 3 is a and t 2, a and t 2, this is n 3. And the output your calling t 3. And n 4 is b and t 1 b and t 1 this is your n 4 and output your calling t 4 and the last gate this is n 5 t 3 and n 4 a generate out right.

(Refer Slide Time: 16:10)

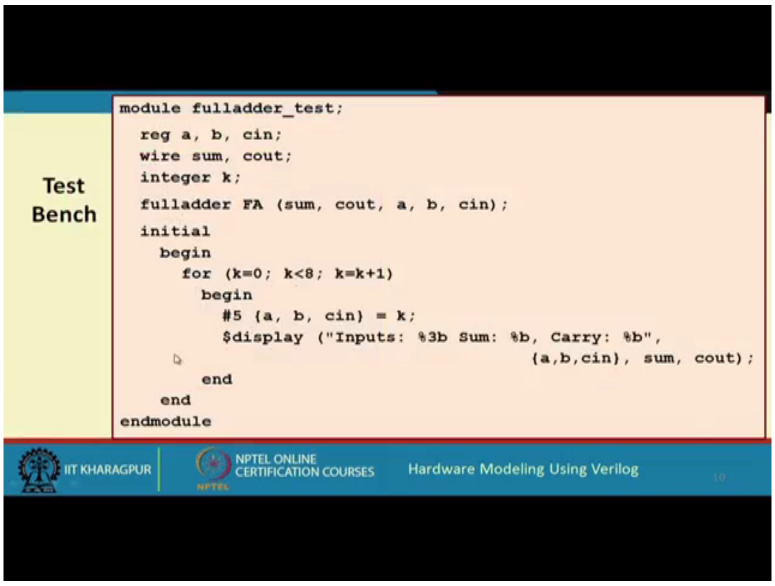


```
module cmosnand (f, x, y);
  input x, y;
  output f;
  supply1 vdd;
  supply0 gnd;
  pmos p1 (f, vdd, x);
  pmos p2 (f, vdd, y);
  nmos n1 (f, a, x);
  nmos n2 (a, gnd, y);
endmodule
```

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Hardware Modeling Using Verilog

So, this is how you design all of them using CMOS nand and of course, the fine final transistor level CMOS nand is this. So, you have done a perfectly hierarchical design with the transistor level to input nand gate as at the leaf, and using that 2 input nand in a hierarchal way you have designed a full adder right.

(Refer Slide Time: 16:30)



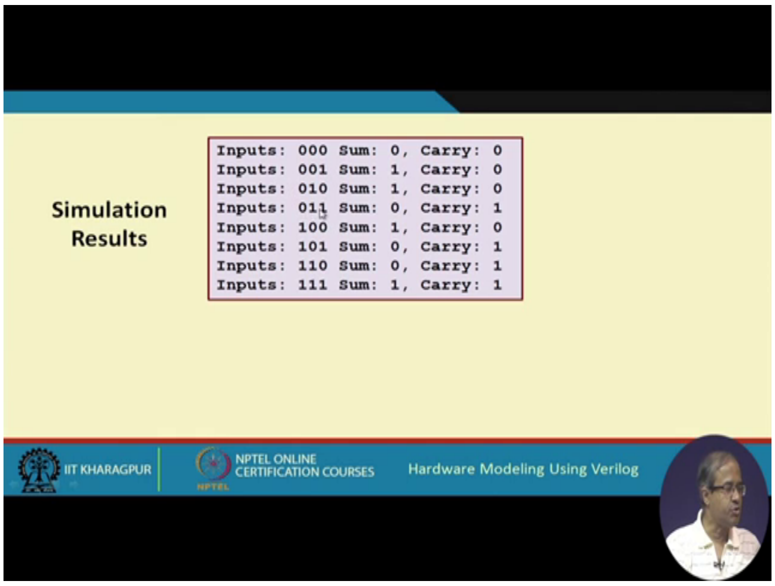
Test Bench

```
module fulladder_test;
  reg a, b, cin;
  wire sum, cout;
  integer k;
  fulladder FA (sum, cout, a, b, cin);
  initial
    begin
      for (k=0; k<8; k=k+1)
        begin
          #5 {a, b, cin} = k;
          $display ("Inputs: %3b Sum: %b, Carry: %b",
                                    {a,b,cin}, sum, cout);
        end
    end
endmodule
```

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Hardware Modeling Using Verilog

So, again we have written a test bench. So, we have instantiated the full adder there are 3 inputs a b c in. So, again we have done this for loop for 7 times generated all values are a b c in, and you are printed the values of a b c in together as a 3 bit number and sum and carry.

(Refer Slide Time: 16:56)

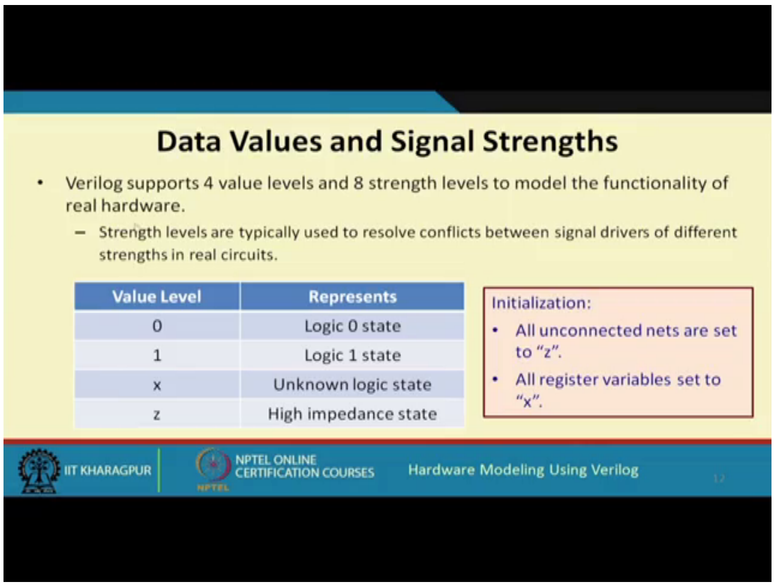

Simulation Results

```
Inputs: 000 Sum: 0, Carry: 0
Inputs: 001 Sum: 1, Carry: 0
Inputs: 010 Sum: 1, Carry: 0
Inputs: 011 Sum: 0, Carry: 1
Inputs: 100 Sum: 1, Carry: 0
Inputs: 101 Sum: 0, Carry: 1
Inputs: 110 Sum: 0, Carry: 1
Inputs: 111 Sum: 1, Carry: 1
```

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES Hardware Modeling Using Verilog

So, when you are simulated the output is coming like this. Let say one let say 0 1 one if we add them sum is 0 carry is 1 1 1 1 sum is one carry is 1, you can verify the others 0 0 1 sum is one carry is 0. So, it is giving correct result fine.

(Refer Slide Time: 17:15)


Data Values and Signal Strengths

- Verilog supports 4 value levels and 8 strength levels to model the functionality of real hardware.
  - Strength levels are typically used to resolve conflicts between signal drivers of different strengths in real circuits.

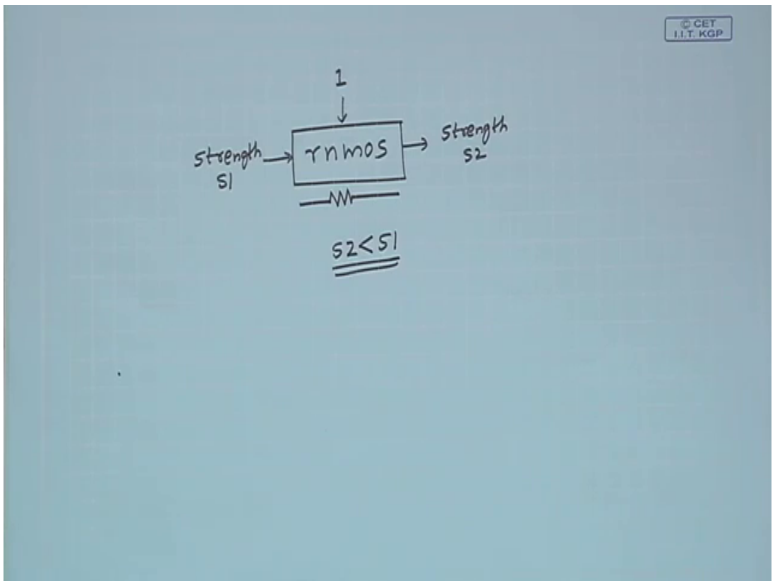
Value Level	Represents
0	Logic 0 state
1	Logic 1 state
x	Unknown logic state
z	High impedance state

Initialization:
- All unconnected nets are set to "z".
- All register variables set to "x".

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES Hardware Modeling Using Verilog

Now, we have seen that how we can use this kind of switch level primitives the transistors, CMOS switches to design low level circuits gates and using gates you can design more complex things of course. But as I said means if you are really into designing circuits using low level CMOS or mos transistors, you need to understand the concept of signal strength very clearly. Well, here as part of this course we shall not be going to the very details of the signal strength, but what I told you earlier means I am just trying to say the same thing. Suppose I use a primitive here which is let say rnmos resistive NMOS ok.

(Refer Slide Time: 18:03)

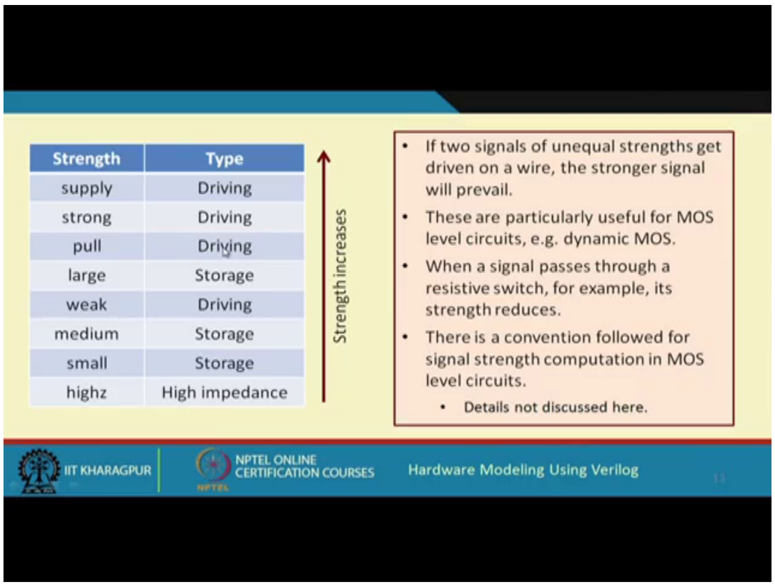


So, there will be a gate there will be a source and a drain. So, this acts as switch. Suppose I have a signal here let say the input I have a signal of strength, let say s 1. And this switch is on in the control input I have applied logic one. So, the switch is on. So, the output I am getting another signal which is of strength s 2. So, because this is resistive. So, even when this is on there will be a resistance in series this s 1 this s 2 value strength of s 2 will be less than the strength of s 1. There will be a degradation in strength you have to remember this thing right.

So, this is something which also we mentioned earlier. So, in verilog there is a 4 logic levels supported you already know 0 1 x and z. And 8 signal levels, this 8 levels are used to model the various kinds of scenarios for real hardware. Like you think the strongest signals will be the one which are directly connected to the power supply lines. If you take a signal directly from v d d or ground those will be the strongest signals, you take the output of a gate strength will be less you take the output from a mos switch which is resistive this strength will be even less, you take a pull up or pull down strength will be even less.

So, there are various circuit design scenarios where you can identify various different signal levels and depending on that you can actually do a calculation of the signal level the simulation or the simulate or does exactly that simulate or understands that how signal strength values are degraded, and how they get modified as they passed through different circuit modules, and they will provide you the calculation of the simulation result accordingly right.

(Refer Slide Time: 20:36)



Strength	Type
supply	Driving
strong	Driving
pull	Driving
large	Storage
weak	Driving
medium	Storage
small	Storage
highz	High impedance

Strength increases

- If two signals of unequal strengths get driven on a wire, the stronger signal will prevail.
- These are particularly useful for MOS level circuits, e.g. dynamic MOS.
- When a signal passes through a resistive switch, for example, its strength reduces.
- There is a convention followed for signal strength computation in MOS level circuits.
    - Details not discussed here.

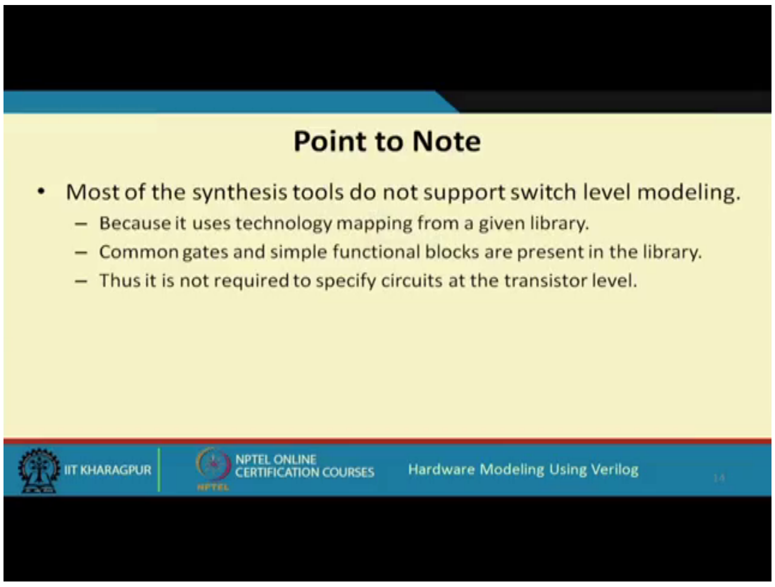
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Hardware Modeling Using Verilog

So, this is something again which we mentioned these are the 8 signal values which are supported. Supply has the highest strength as I said then you say strong, pull, large, weak, medium, small. And high impedance high impedance has the lowest strength high impedance is something which is like floating. A wire floating I just a float wire in my hand I say what is the voltage here there is no voltage because it is not electrically connected to anything.

So, the signal strength of the high impedance signal or a line or a variable will be the lowest and the supply values one and 0 that will be the highest. And there are many I means intermediate values, there are some strength which are used for storage some used for driving other gates. So, I am not going to the details of this, but the point to notice that I also mentioned this earlier if 2 signals of unequal strengths drive a single wire then the stronger signal will prevail. Suppose there are 2 modules the output of the 2 module is driving a single wire, but the output signals they have different strengths.

So, the signal which has highest strength, that will over ride the other and the output value will get it, but if they are of the same strength and their values are conflicting then the output will become indeterminate x fine. So, for mos level circuit modeling this multiple values of signal strengths are useful. Like I will also mentioned this that when a signal will pass through a resistive switch it is strength will reduce. But the details as I said details of signal competition the kind of conventions that are followed, that we have not discussed here.

(Refer Slide Time: 22:49)



Point to Note

- Most of the synthesis tools do not support switch level modeling.
  - Because it uses technology mapping from a given library.
  - Common gates and simple functional blocks are present in the library.
  - Thus it is not required to specify circuits at the transistor level.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Hardware Modeling Using Verilog

So, the point to note finally is that well we use transistor level modelling for mainly simulation. For those of us who are into transistor level design who have design a circuit at the level of transistors, who want to simulate and see that well my design is correct or not you can use these facilities which are provided in verilog for switch level modelling to do that, but if you target is for synthesis. For example, if you are doing an a p j mapping see a p j there is nothing like mos there is nothing like mos transistor that everything you can map into are some modules and blocks. There call c l b is combination logic blocks some flip flops and so on.

So, synthesis tools will not be accepting any design which are at the level of the transistors. So, this is true not only for a p js, but also for a 6. For a 6 I told you typically the designs are created by picking up the cells. From a technology library and the technology library already has some very well optimised transistor level layouts. The synthesis tool will not allow you to manipulate on that and give a different net list of the transistor level of course, let me tell you there are some designs where you may have to design something at the level of transistors. You may have to create a layout, but there are separate low level tools and software available for that. You do not use a high level synthesis tool that takes verilog and translates in to hardware for that purpose ok

So, most of the synthesis tools not most means almost all of them you can say do not support switch level modelling. Because the final hardware that is generated it is not, generated from the switch level models you have specified they are usually taken from a library ok.

So, circuits at the transistor level is not required to be specified if your final target is synthesis all right. So, with this we come to the end of this lecture. So, we have seen very briefly the feature that is available in the verilog language to model circuits at the level of mos transistors, this is called switch level modelling.

Now, in the next lectures we shall be taking a case study a fairly complex case study of a processor. And you shall see how you can implement it using a pipeline fashion using verilog.

Thank you.