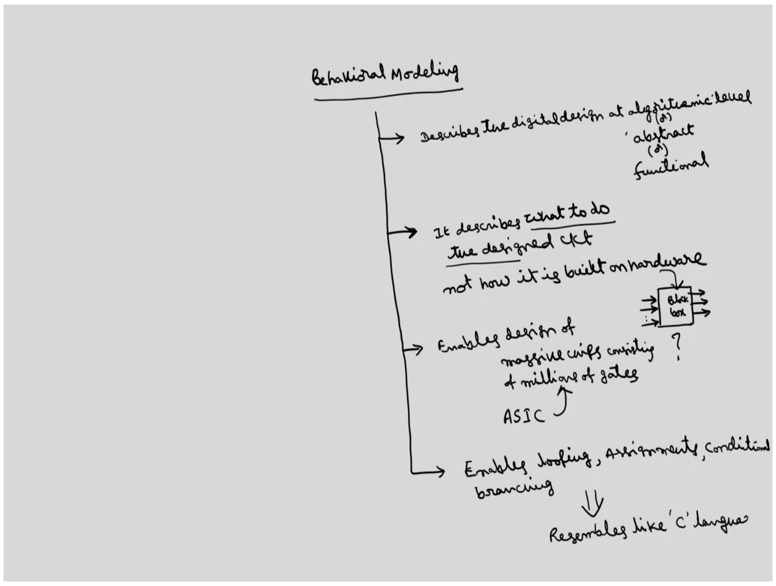
# System Design using Verilog

**Dr. Shaik Rafi Ahamed**
**Department of Electronics and Electrical Engineering**
**Indian Institute of Technology, Guwahati**

## Dataflow and behavioral modeling
## Lecture - 16
## Basics of behavioral modeling

Ok, in the last lectures, we have discussed about the structural or gate level modeling and then switch level or transistor level modeling, then Dataflow modeling and the last level of design description is behavioral modeling.

(Refer Slide Time: 00:53)



Behavioral Modeling

- Describes the digital design at algorithmic level (or) abstract (or) functional
- It describes what to do the designed ckt not how it is built on hardware
- Enables design of massive chips consisting of millions of gates
  ASIC
- Enables looping, Assignments, conditional branching
  $\Downarrow$
  Resembles like 'C' language

So, today we will discuss about the behavioral modeling. So, normally if the digital circuit consisting of fewer gates; so, a convenient way to model is by using structural or gate level modeling, whereas if the number of gates is more, then we have to go for the behavioral modeling.

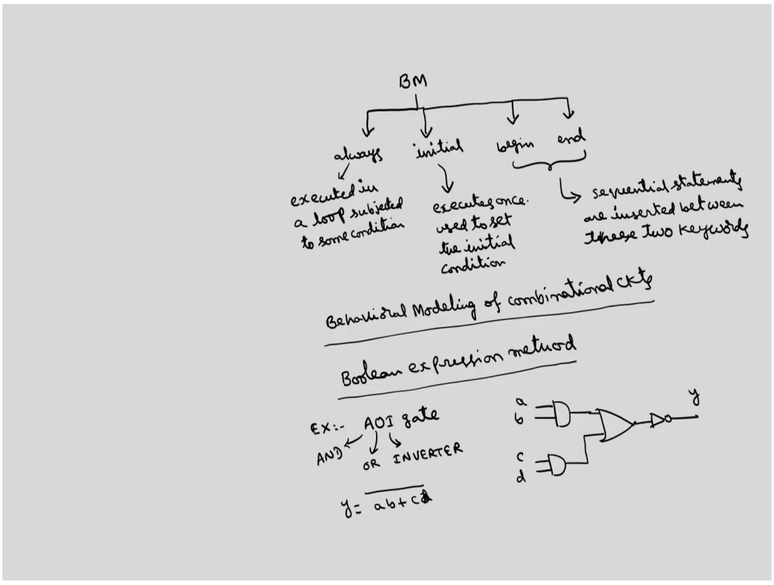
So, this behavioral modeling basically describes the digital design at algorithmic level or you can also call it as abstract level or functional level. This is the high-level modeling. So, basically this behavioral modeling will describe the digital circuit in terms of the function. So, that is, it describes what to do the designed circuit, not how it is built on hardware.

So, we do not bother about the internal hardware of the circuit, just we will consider as a block box; so this is block box we will give some inputs, then we will have some outputs.

Then in behavioral modeling, we will discuss what the function that will be performed by this block box, without knowing the hardware inside this block. So, because of that, this enables design of massive chips, consisting of millions of gates. So, that is why in today's industry, so mostly we will use behavioral modeling; because the industrial applications specific integrated circuits, they consist millions of the gates.

So, in order to describe this behavior of this ASIC designs, so a convenient way is use of behavioral modeling. Behavioral modeling actually, it enables the looping, it also enables the assignments, the conditional branching. So, because of these features, this resembles like C language program. So, we discussed about the various keywords used in the behavioral modeling while discussing about the introduction of Verilog lecture.

(Refer Slide Time: 05:35)



BM

always — executed in a loop subjected to some condition

initial — executes once. used to set the initial condition

begin end — sequential statements are inserted between these two keywords

Behavioral Modeling of combinational ckts

Boolean expression method

EX:- AOI gate

AND OR INVERTER

$y = \overline{ab + cd}$

So, this behavioral modeling basically I will call as BM behavioral modeling. So, this basically uses the keywords always; this is similar to the dataflow modeling, initial, begin and end.

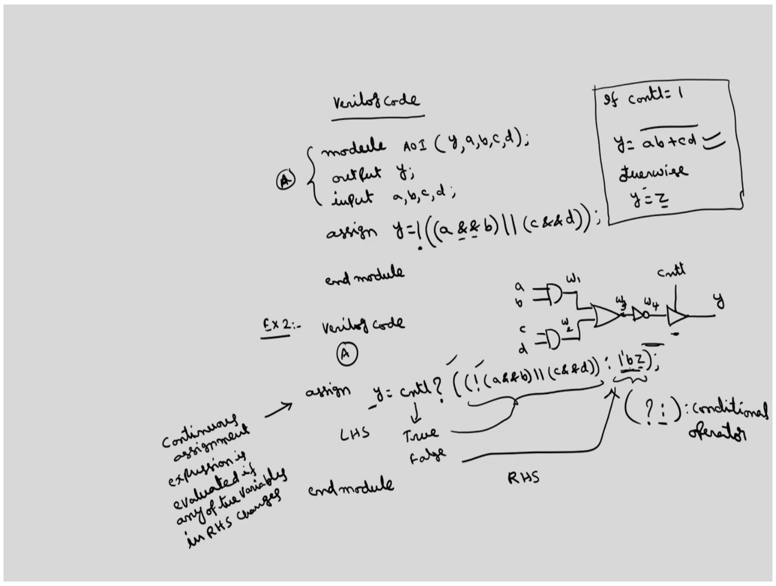
So, always as the name implies, so this will be executed in a loop subjected to some conditions; whereas this initial as the name implies, it executes only once. So, it will be normally used for setting the initial condition. Like if I take the ripple carry adder, what is the initial carry?

If you want to set initial carry as 0; so, we can initiate that $c_0$ is equal to 0 using this initial block. And there are two more blocks begin and end, normally whatever these sequential statements will be written between this begin and end, between these two keywords or blocks.

So, to start with the examples, first we will discuss the combinational circuit modeling using behavioral modeling of combinational circuits. First I will discuss using Boolean expressions, combinational circuit described in terms of Boolean expressions.

If I take example of AOI gate; this stands for AND gate, this stands for OR, this stands for INVERTER. Let the gate is say for example, we have two AND gates, one OR gate followed by one NOT gate; say output is y, inputs are a, b, c, d. So, the Boolean expression for this one is y is equal to (a b + c d) bar. Then how to model this using behavioral description?

(Refer Slide Time: 09:45)



Verilog code

```
(A) { module AOI (y,a,b,c,d);
    { output y;
    { input a,b,c,d;
      assign y=!((a && b)||(c&&d));

      end module
```

If contl= 1

$y = \overline{ab+cd}$

otherwise

$y = z$

Ex 2:- Verilog code

(A)

```
assign y= cntl? ((!(a&&b)||(c&&d)) : 1'bz);

end module
```

LHS

True

False

RHS

$( ? : )$ : conditional operator

Continuous assignment expression is evaluated if any of the variables in RHS changes

So, there are four inputs and one output. So, I will call this module, I will give some name AOI; output is y, inputs are a, b, c, d. Then we need assign statement, assign y is equal to. So, y is actually (a b + c d) whole bar; a we have to AND with b. Here we can use single AND operator or you can use two operators, if they are single bit numbers.

If they are multiple bit numbers, then it will be different, ok. So, here we can use a single AND operator or we can use two AND operators; because a, b, c, d are single bit numbers. So, this is 1 bit number. So, all are 1 bit numbers.

So, we can use either && or single & it is same. Similarly ||, | it is same if the variables are single bit; c AND d this is NOT. So, the entire Boolean expression is represented by this statement, then end module; this is a simple Verilog code corresponding to the Boolean expression.

Suppose if I want to check some condition; so, in the previous circuit, if I take a second example, where I am going to add a buffer to this. So, if I use the same circuit, I am going to add a buffer with control signal.

So, the output will be this expression, y is equal to (a b + c d) whole bar if control is equal to 1; otherwise because this is a inverter, this is what we have to implement for this circuit. So, the control signal is going to decide whether the output is the original expression (a b + c d) whole bar or output is z; z is nothing but high impedance state.

So, we know that this buffer will be having three states; logic 0, logic 1, and high impedance state. In high impedance state, this will act as an open circuit, so this will not draw any current. So, if you want to write the Verilog code correspond to this; so, these three statements are same, we call this as some A.

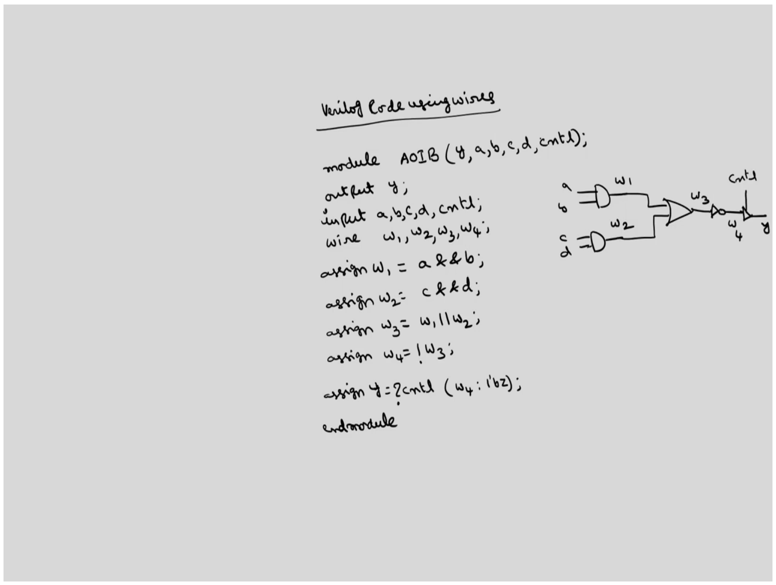
So, A here all the three statements are same, then we can use a conditional statement. So, we know that assign output y = CNTL?. So, we know that this is the conditional operator, which you have discussed in earlier classes. So, what are the two conditions you have to check? The output can be if the control is logic 1; the first expression is executed if control is logic 0, second expression will be executed, these two has to be separated by colon.

The same, this colon 1’b z. So, we know that this statement of this 1’b z, 1 stands for 1 bit, b is binary, and the value is z. So, if this control signal is true, the first expression is executed, this one is executed; if it is false, then the second condition, this is executed then end module.

So, this assignment is actually called as continuous assignment. When does this RHS, right hand side will be assigned to the LHS left hand side which is y? If any one of these variable changes; continuous statement, this particular expression is executed, expression is evaluated if any of the variables in RHS changes.

So, whenever any variable changes, this will execute this expression and then it will be assigned to y, that is why the name continuous assignment. We can also write this program using wires. So, instead of writing single expression here, I can define here wires w1, w2, w3, w4.

(Refer Slide Time: 17:01)



Verilog Code using wires

```
module  AOIB (y, a, b, c, d, cntl);
output y;
input a, b, c, d, cntl;
wire  w1, w2, w3, w4;
assign w1 = a && b;
assign w2 = c && d;
assign w3 = w1 || w2;
assign w4 = !w3;
assign y = ?cntl (w4 : 1'bz);
endmodule
```

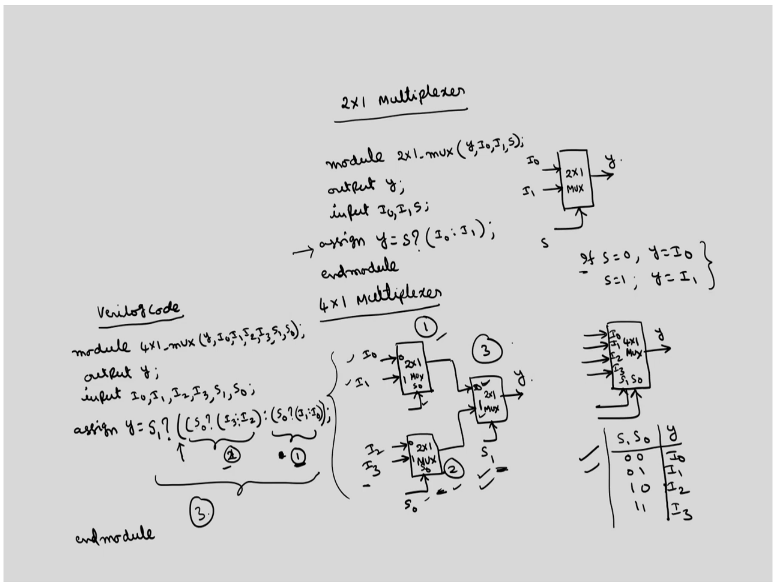
a
b
c
d
w1
w2
w3
cntl
w4
y

If you want to write using wires the same program; module, then call as a AND OR Inverter and Buffer also. Output is y, inputs are a, b, c, d; output is y and one more input is there enable which is control, a, b, c, d, cntl; output is y, inputs are a, b, c, d, control and there are four wires w1, w2, w3, w4.

So, first I am going to assign w1. So, what is w1? This is ab, this we have called as w1. So, w1 is equal to a AND b. Similarly, w2 is c AND d, then w3 is w1 OR w 2, then w 4 is NOT of w 3.

Then the final output depends upon control. So, you can write the same conditional statement, assign y is equal to ?cntl. So, it can be either z or w4; w4 if it is true, otherwise 1'b z, then end module. So, you see the Verilog code of this AND, OR, Invert, Buffer gate using wires.

(Refer Slide Time: 20:21)



2x1 Multiplexer

```
module 2x1_mux (y,I0,I1,S);
output y;
input I0,I1,S;
→ assign y = S? (I0:I1);
endmodule
```

If S = 0, y = I0
S = 1; y = I1

4x1 Multiplexer

Verilog Code

```
module 4x1_mux (y,I0,I1,I2,I3,S1,S0);
output y;
input I0,I1,I2,I3,S1,S0;
assign y = S1? ((S0? (I3:I2)):(S0?(I1:I0));
endmodule
```

| $S_1 S_0$ | y |
|---|---|
| 0 0 | $I_0$ |
| 0 1 | $I_1$ |
| 1 0 | $I_2$ |
| 1 1 | $I_3$ |

I0
I1
2x1 MUX
y
S
I0
I1
I2
I3
2x1 MUX
S0
S1
① ② ③
4x1 MUX
S1 S0

Now, we will discuss the other combinational circuits, first I will discuss the multiplexer, 2 by 1 multiplexer. So, we know that for 2 by 1 multiplexer, 2 inputs I0, I1, one selection signal S, one output y. If S is 0, output should be I0; if S is 1, output y is I1, this is if else structure. So, this if else structure will be executed by always block module; this of course we have already discussed in the earlier classes.

2 by 1 mux, output is y, inputs are I0, I1, selection; output y, input I0, I1, S; assign output y is equal to S condition. I think in the previous, this condition should be after this; S? I0:I1, end module. So, we know that in gate level modeling to model the higher order multiplexers; we realized this higher order multiplexers in terms of the lower order multiplexers and then we have instantiated.

So, the similar type of the concept can be used for the behavioral modeling also. Suppose if you want to implement 4 by 1 multiplexer; here we have to use four conditions. So, like this here we have two selection signals S1, S0, four inputs and one output y. So, S1, S0 is going to decide the output y; 00, I0; 01, I1; 10, I2; 11, I3. So, you have to write this assign four such statements, instead of that using single assign statement only we can write this.

So, for that we have to realize this 4 by 1 multiplexer in terms of 2 by 1 multiplexers. So, as we have discussed in the earlier lectures. So, this 4 by 1 multiplexer can be implemented in terms of three 2 by 1 multiplexers in this manner; this is I0, I1, this is connect to 0,1; there is another 2 by 1 multiplexer, this is I2, I3, internally this is 0th and 1st.

Then the selection signals of these two will be same as S0 then these two will be applied to another 2 by 1 multiplexer; this is 0,1 this will be S1, this is final output y. You can easily verify this truth table will be obtained for this circuit.

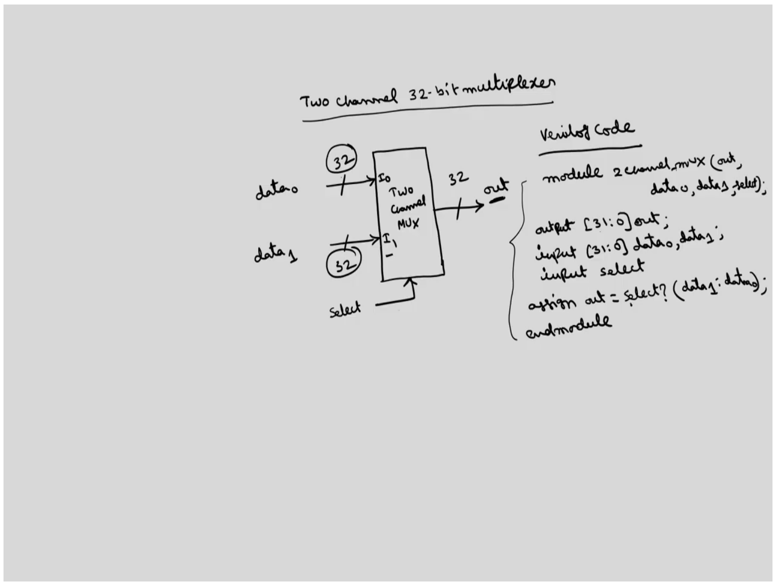
Now, we can use only single assign statement to implement this 4 by 1 multiplexer using this. So, what will be the code corresponding to this one? Module 4by1mux y, I0, I1, I2, I3, S1, S0; output y, input I0, I1, I2, I3, S1, S0. Then single assign statement, assign y is equal to S1; we are going to first find out this S1 condition, if S1 is equal to 1.

So, 0 this has to be selected, 1 this has to be selected. So, this one has to be selected I2, I3; I0, I1 has to be selected. So, this is 1; 1 means this has to be selected; S0, this is also S0. 1 means I3, 0 means I2:, so, this is corresponding to, if I name these multiplexers as 1, 2, 3; so this is correspond to 1 and 2 is in a similar way we have S0 only, because the selection lines of these two are same, but the inputs are for 1 it is I1, for 0 it is I 0.

So, this is for, this is for 2, this is for 1, and this entire thing is for 3. So, this parenthesis started here, it will end here, then semi colon, this is for 3. So, using single assign statement; so this is nothing but instantiating this 2 by 1 multiplexer and then this instantiating this 2 by 1 multiplexer ok to get overall the multiplexer, end module.

So, this is how we can use for the larger multiplexers; we can instantiate smaller multiplexer in a similar way to that of the structural or gate level modeling. Next, we will discuss about two channels 32-bit multiplexer.

(Refer Slide Time: 28:19)


Two channel 32-bit multiplexer

Verilog code

```
module 2channel_mux (out,
            data0, data1, select);
output [31:0] out;
input [31:0] data0, data1;
input select
assign out = select? (data1: data0);
endmodule
```

That is, we have two channels, each channel consisting of 32-bit numbers; this is data 0, this is data 1, this is 32-bit number, this is also 32-bit number. And we have only one select signal, this is two channel mux. Output is again 32-bit number, we can call this as out; say either of this 32-bit numbers will be transferred to the output selected by this selection signal, this is I0, say this is I1.

So, now here in the previous multiplexer we have a scalar, now we have vector. So, we can define the vectors; we know how to define the vectors also that we have discussed in the earlier lectures. So, the code corresponds to this one is module 2 channel mux out, data 0, data 1, select.

Output [31:0] out; this out is the signal name, this is 32-bit. Input [31:0] data 0, data 1; then another input is select. Then assign out is equal to select data 1, because for true? data 1, else data 0 then end module.

So, if it is gate level, it will be much more complicated; because, so if we have a 32-bit numbers, two 32-bit numbers you have to multiplex, so it will take lot of circuitry. It will require complex circuitry; so, you have to explain in terms of the logic gates in case of structural level modeling. So, it will become complicated, whereas, if I use the behavioral modeling; so, we can write using only a simple Verilog code.

So, next we will we have some other combinational circuits, such as decoders and then we will discuss a small ALU also, sequential circuits. So, all those things we will discuss in the next lecture.

Thank you.