System Design through VERILOG

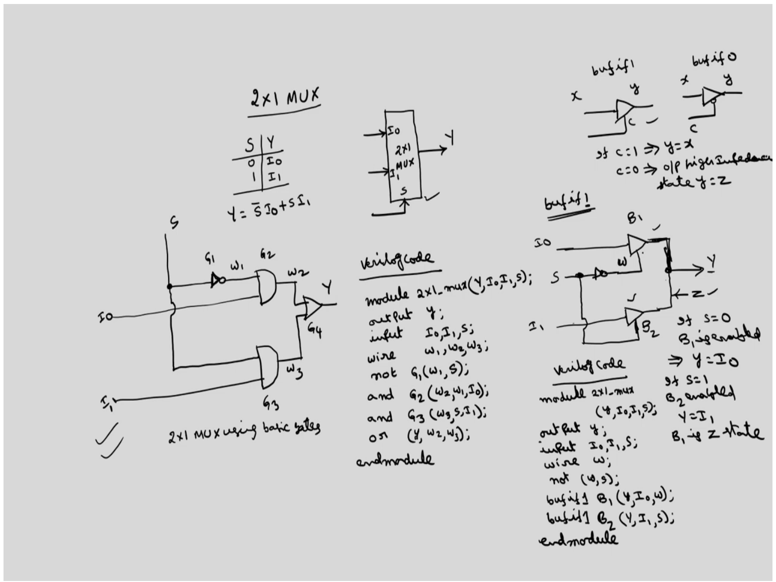
Prof. Shaik Rafi Ahmed
Department of Electrical and Electronics Engineering
Indian Institute of Technology, Guwahati

Gate level modeling - II
Lecture - 08
Demultiplexer, Read Only Memory

In the last class, we were discussing about the Multiplexer design, after that we have discussed about the Verilog code. So, normally we can start with the lower size of the multiplexer and we can construct the higher size of the multiplexers by instantiating this the lower order multiplexers, ok.

So, yesterday I have taken example of a 4 by 1 multiplexer and we have discussed circuit diagram and then Verilog code. For constructing the larger from the smaller, I will start with the smaller possible multiplexer which is 2 by 1 multiplexer.

(Refer Slide Time: 01:10)



Verilog code

```
module 2x1_mux(Y,I0,I1,S);
output Y;
input  I0,I1,S;
wire   w1,w2,w3;
not  G1(w1,S);
and  G2 (w2,w1,I0);
and  G3 (w3,S,I1);
or   (Y, w2,w3);
endmodule
```

Verilog code

```
module 2x1_mux
          (Y,I0,I1,S);
output Y;
input I0,I1,S;
wire w;
 not (w,S);
 bufif1 B1 (Y,I0,w);
 bufif1 B2 (Y,I1,S);
endmodule
```

So, we can design in a similar manner to that of 4 by 1 multiplexer. So, this 2 by 1 multiplexer will have two inputs as the name implies one output I0 I1 or the two inputs and Y is the output and this will have one selection line, S is the selection line. So, and the operation of this 2 by 1 multiplexer is S is going to decide which input is going to be transferred to the output. If S is 0, output becomes I0; if S is 1, output is I1.

So, the Boolean expression for this will be y is equal to S bar I0 plus S I1. So, the circuit diagram of 2 by 1 multiplexer will be. So, we have take S we have to apply through the NOT gate then another input is I0 other input is I1 and this S we have to apply directly.

So, these two we have to OR. This is output y, this is I0 I1 and this is selection signal S. This you call as some wire W1 here also we require W2 wire W3 wire ok. So, it is the verilog code corresponding to this one. module I am giving the name as 2 by 1 under score mux. So, the overall input and outputs of this multiplexer are I0 I1 are the inputs, S is the control signal output is Y.

So, Y comma I0, I1 and S. output y, input I0 I1 and S there are 3 wires are required W1, W2 and W3. Then NOT to obtain this wire W1, output is W1, input is S, then AND you can give the name as G1 G2. The output of G2 is this AND gate if I call this as G2. G1 G3 G4. The output of G2 is W2, inputs are W1 and I0 and for the AND gate G3, output is W3, inputs are S and I1. Then finally, the output y inputs are W2 and W3. This is the end module. This is a simple Verilog code for a 2 by 1 multiplexer.

So, we can construct the larger multiplexers using 2 by 1 multiplexers. So, before that so, I will do some other realization this is using basic gates 2 by 1 mux using basic gates whereas, 2 by 1 multiplexer can also be constructed using the buffers ok. So, we have already discussed a buffer is nothing, but a current amplifier.

So, this will be something like the input x, output y some control signal c. If c is equal to 1, output is equal to input. If c is equal to 0, output is in high impedance state that we will call as z, y is z. This is the case of a positive buffer. We can have the reverse way also c is equal to 0, output is equal to input; c is equal to 1, output will be high impedance state.

So, this buffer is normally referred by buffer if 1 means if control signal is 1, output is equal to input; if control signal is 0, output will be in high impedance state. On the other hand, we can have buffer like this one, we can have a bubble here. So, c is S, x y the operation is reverse. For this we will call as buffer if 0, this is buffer if 1.

So, using this type of buffers also you can construct a 2 by 1 multiplexer. So, how to construct is, so, I will take just simply two buffers are enough. I will take buffer if 1 the first step of the buffer. This I will give as I0, this I will give as I1, the selection signal I will give this is selection signal through NOT gate to this control signal and this I will give without inverter to the selection or the control signal of the second buffer.

But, multiplexer will be having only one output. So, these two I am going to join together. This is output y. We can easily see the operation of this one is this is buffer if one type of thing means if this control signal is 1, output is equal to input. So, if S is equal to 0 so, what happens the upper buffer will be enabled because this will become 1, the lower buffer will be disabled ok.

So, if I call this 1 as B1 B2, B1 is enabled means output of the B1 is this output itself is y. So, y becomes I0 and what about this output of this buffer because this control signal is 0 output will be in high impedance state this is output is in z high impedance state means almost this will be open circuited this will be cut; high impedance means it draws almost 0 current. So, we can assume that this is a open circuit type of thing.

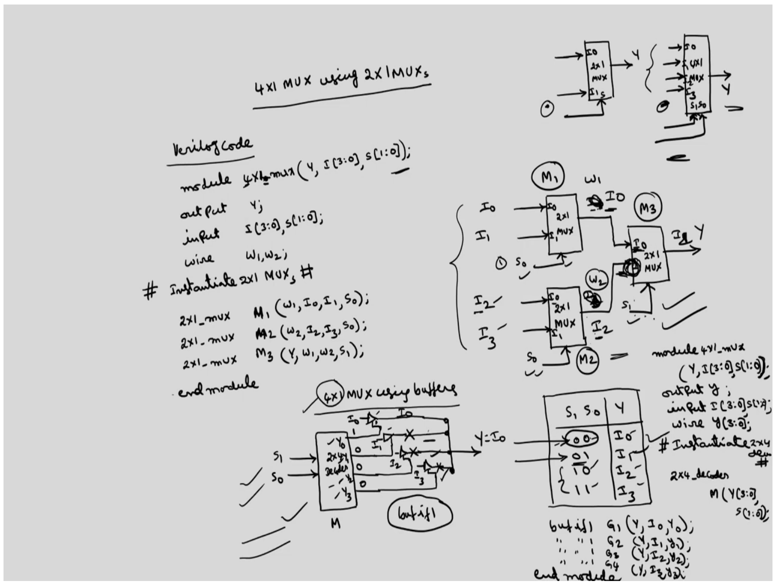
On the other hand, if S is equal to 1, reverse, B2 is enabled and B1 is in high impedance state. So, output is equal to I1, B1 is in high impedance state. So, this is a simple circuit using only two buffers and one NOT gate also you can construct the multiplexer. So, whereas, here we require more gates. So, we require two 2-input AND gates, one 2-input OR gate, one NOT gate. Here simply we require two buffers and one NOT gate. This is an efficient implementation.

So, for this what will be the Verilog code? Module 2 by 1 mux, y is the output, I0 I1 are the inputs and S is the selection signal same as the previous one and output is y, inputs are I0, I1 and S. What are the wires required here? Is only one wire is required this if I call this as W which is S bar. wire W, NOT W comma S buf if 1, I will call this as B1.

So, what is the output and what are the inputs? Output is y, inputs are I0 and W. Buf if 1 we can construct with 0 also ok this is buf if 1, B2 the outputs are y, inputs are I1 and S. This is simple code, this is end module. So, using this we can generate or we can implement a 2 by 1 multiplexer ok. Now, we have seen the implementation of 2 by 1 multiplexer, then how to construct a 4 by 1 multiplexer using 2 by 1 multiplexers.

Using logic gates we have discussed in the last class ok. Now, how to implement a 4 by 1 multiplexer using 2 by 1 multiplexers? First of all how many minimum number of 2 by 1 multiplexers are required to construct a 4 by 1 multiplexer.

(Refer Slide Time: 11:42)



```
module 4x1_mux(Y, I[3:0], S[1:0]);
output Y;
input  I[3:0], S[1:0];
wire   W1,W2;
# Instantiate 2x1 MUXs #
2x1_mux  M1 (W1, I0, I1, S0);
2x1_mux  M2 (W2, I2, I3, S0);
2x1_mux  M3 (Y, W1, W2, S1);
endmodule
```

S1 S0	Y
00	I0
01	I1
10	I2
11	I3

```
module 4x1_mux
   (Y, I[3:0], S[1:0]);
output Y;
input I[3:0] S[1:0];
wire Y[3:0];
# Instantiate 2x4 dec #
2x4_decoder
      M (Y[3:0],
         S[1:0]);
bufif1 G1 (Y, I0, Y0);
 "  "  G2 (Y, I1, Y1);
 "  "  G3 (Y, I2, Y2);
 "  "  G4 (Y, I3, Y3);
endmodule
```

So, we know the block diagrams of both the 2 by 1 and 4 by 1 multiplexers. This is 2 by 1 multiplexer with 2 inputs I0 I1 and one output Y and one selection signal. It has 4 by 1 multiplexer will be having 4 inputs, 1 output, 2 selection signals. This is S1 S0 if output Y 4 by 1 mux I0 I1 I2 I3. It is clear that if you want to construct a 4 by 1 multiplexer ok so, this 4 inputs are required here at the input here this is having only 2 inputs.

So, we require at the first stage we require 2 such 2 by 1 multiplexers. So, that the first 2 by 1 multiplexer will give 2 inputs; this is one 2 by 1 multiplexer this will give 2 inputs I0 I1 and we require one more 2 by 1 multiplexer. This will contain 2 more inputs, inside this I will call as I0 I1 only; outside I will call this one as I0 I1 this I will call as I2 I3.

So, now, the number of inputs here are 4 here also 4, this is same, but each multiplexer will give one one output each, but I want only final one output only ok. So, for that we require 1 more 2 by 1 multiplexer. This is I0 I1 2 by 1 mux this is also having another selection line, this is also having single selection line, this is also having single selection line and finally, we will have one output final output Y.

This is the basic structure of constructing this 4 by 1 multiplexer using 2 by 1 multiplexer here we do not require any external gate not even NOT gate, but we require 3 such 2 by 4 multiplexers ok. Now, the question is how to select this selection signals. So, what is for 4 by 1 this one S1 S0 are the input control signals output should be for 0 0 it should be I0; 0 1, I1; 1 0, I2; 1 1, I3. So, I want to implement this table.

So, what should be the values of this control signal such that this table will be implemented by using this circuit ok? So, you can see that for S is equal to 0, S 0 is equal to 0, S 1 is equal to 0, I want I0. So, this you can easily I mean by seeing this we can say that this is S0, this is also S 0, this you take S1. So, S0 for the first combination S0 is 0.

So, this I0 is selected here because S0 is 0, here I0 is selected at I0 we have connected I2. So, I2 is selected because S1 is also 0, I0 of this one is. So, final output is I0 only whereas, if I take the second combination S1 is 0, S0 is 1, S0 is 1 means if this is 1, I1 is selected here, this will be having I1 this will be having I3, but still S1 is 0, S1 is 0 means I0 will be selected what is connected to I0 I1. So, output will be I1.

Similarly, when S1 is equal to 1 always this will be selected for these two combinations only this will be selected ok whereas, here S0 is 0 in this combination S0 is 1 in this combination. When S0 is 0, so, this 0, I0 will be selected here I2 will be selected, but because this S1 is 1 in both the cases this will be selected what is that I2. So, output will be I2. Similarly, when S0 is equal to 1 so, this output will be here I3 output will be here will be I1, but only this is selected. So, output will be I3.

So, this will show the implementation of a 4 by 1 multiplexer using three 2 by 1 multiplexers ok. So, now, how to write the Verilog code corresponding to this one? So, overall module of this 4 by 1 multiplexer is this. We have written this Verilog code for this 4 by 1 multiplexer using basic gates whereas, here using three 2 by 1 multiplexers and we are going to instantiate this.

So, module 4 by 1 underscore multiplexer the outputs of this multiplexer are Y, the inputs are I[3:0], S[1:0]. output Y, input I[3:0], S[1:0]. So, there are two wires required. So, I will call this wires as here this I will call as W1 wire, this I will call as W2 wire. So, wire W1 comma W2. So, I have to instantiate now 2 by 1 multiplexers. I will call these modules as M1, this as M2, this as M3.

So, the name that I have given for this 2 by 1 multiplexer is instead of 4 we have 2 here. So, 2 by 1 underscore mux is the name module M1. So, what are the inputs and outputs of this multiplexer? Output is wire W1, inputs are I0 I1 and selection signal S0. Similarly, second 2 by 1 multiplexer M2, output is W2, inputs are I2 and I3. These are external inputs and selection is S0 only same S0.

Then the third 2 by 1 multiplexer M3 output is the final result that we want Y inputs are W1 comma W2 and selection signal is S1. This is end module. This is how you can construct the larger multiplexers using the lower order multiplexers. Similarly, you can go for the higher order 4 by 1 multiplexer if you want. If you want 8 by 1 multiplexer you can use 4 by 1 multiplexers.

So, before going for that I can show that this 4 by 1 multiplexer can be implemented by using the logic gates that we have discussed in the last class and here 4 by 1 multiplexer can also be implemented by using the buffers like 2 by 1 multiplexer. So, how to implement 4 by 1 multiplexer using buffers? Here we require 4 control signals.

So, we can generate this 4 control signals using a 2 by 4 decoder. This will be having 4 output. Let us assume that this is positive logic. So, you take 4 buffers, this is one buffer which control this, this is another buffer which is controlled by this, third buffer which is controlled by the output of the third this one.

So, the inputs of this one are now this is I0, this is I1, this do not connect it, this is I1, this is I2, this is I3 and selection signals of the control signals of this buffers we have connected through this output of the decoder Y0 Y1 Y2 Y3 and these four outputs of the buffers you have to connect together and you call this final output as Y. This is S1 S0 this is the implementation of 4 by 1 multiplexer using buffers.

So, the operation is simple ok. If S1 S0 are 0 0 then Y0 is 1 remaining all are zeros which you have discussed in the earlier classes the operation of the 2 by 4 decoder. So, because these are buf if 1 type of buffers so, this is 1 this selection signal is 1 remaining all are 0s. So, these three will be in high impedance state whereas, this one is I0 because the input of this buffer is I0. So, the output becomes I0 because the remaining 3 connections are broken.

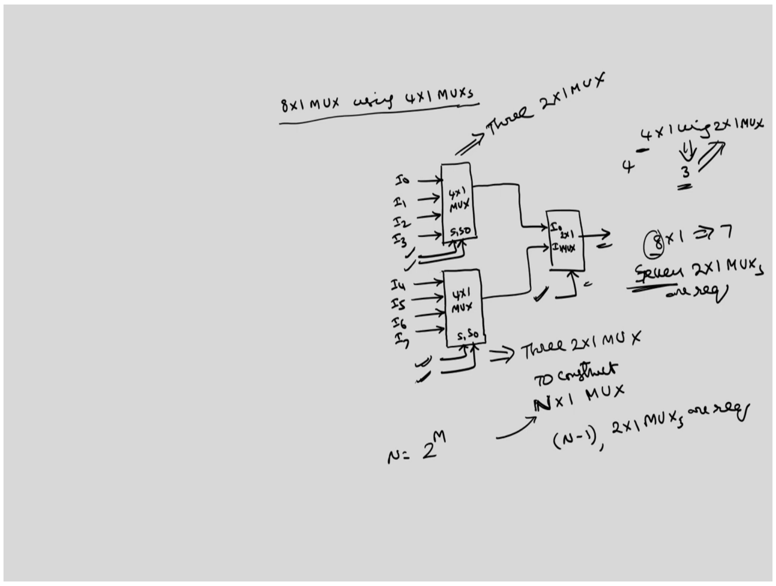
Similarly, when S1 S0 is 0 1 so, only this one will be intact, the remaining three will be almost open circuited high impedance state. For 1 0 only this one is enabled remaining 3 will be broken. For 1 1, only this one will be enabled the remaining three will be in high impedance state. So, as a result of that the output becomes I0 I1 I2 I3. This is a simple circuit where we require only one 2 to 4 decoders and 4 buffers.

So, if you want to write the Verilog code corresponding to this 4 by 1 multiplexer using the decoders we have to instantiate 2 by 4 decoder which we have discussed in the last class. So, if I write Verilog code using this so, you have to define Y0 Y1 Y2 Y3 as the wires. I will write here module 4 by 1 mux the outputs of this one are same Y same as this I[3:0], S[1:0] then output y, input I[3:0], S[1:0] then wires here we require four wires y0 y1 y2 y3, y[3:0]. Then you have to instantiate a 2 by 4 decoder.

So, the name that we have given for the 2 by 4 decoder is 2 by 4 underscore decoder if you assume that this is the name. I can call this one as some module M. So, you can give M module. What are the outputs and inputs? Outputs are Y[3:0], inputs are S[1:0]. So, this will give the 4 wires. Then we have to write buf if, I will write here bufif1 all are 4 buffers are this type of the buffers only.

So, you can call this buffers as G1. So, outputs of the G1 is Y only, input is I0 and control signal is Y0. Similarly, buif1 G2 Y is the output I1 and y1 are the inputs. Similarly, G3, similarly G4 then end module. The outputs of these are Y, I2, y2; Y, I3, y3. This is a code if I want to implement. This 4 by 1 multiplexer using 2 by 4 decoder instead of 2 by 4 multiplexer because this is simple relation.

(Refer Slide Time: 27:19)



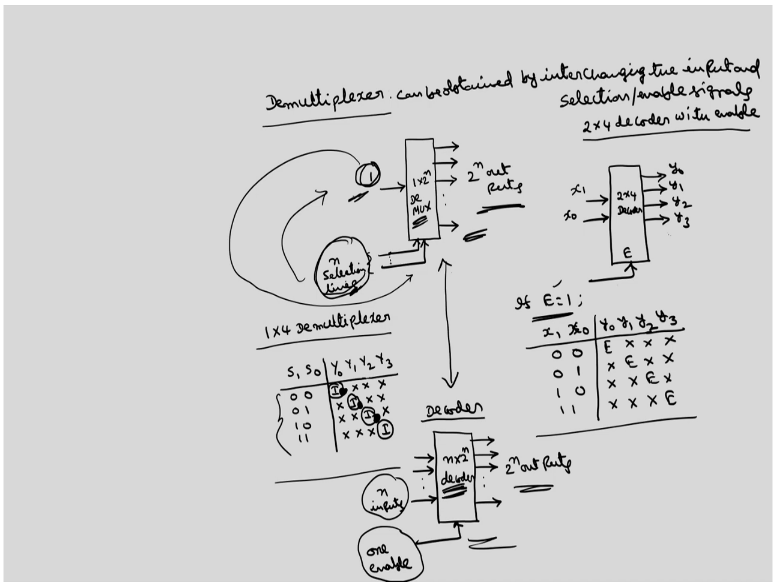
Now, if you want to construct further larger this one suppose if I want to construct 8 by 1 multiplexer using 4 by 1 multiplexers. So, what is the basic construction I want total 8 by 1. So, we require this 4 by 1 will give 4 inputs, this another 4 by 1 will give 4 inputs total 8. This will give 4 this is I0 I1 I2 I3 I4 I5 I6 I7. So, this will give one output this will give another output. So, these two you have to give to some other 2 by 1 multiplexer. This is final output. So, this will be having one selection lines, this will be having two selection lines.

Again this 4 by 1 multiplexer we require three 2 by 1 multiplexers. So, total seven 2 by 1 multiplexers are required. So, depends upon the table you have to choose this selections. As I have explained the logic in the last realization, you have to follow the similar realization and we can have to assign the appropriates this selection signal such that the output will be the output of 8 by 1 multiplexer.

So, we in order to construct 4 by 1 using 2 by 1 muxes we require three such 2 by 1s whereas, to construct 8 by 1 we require 7. So, in general if I want to construct some N by 1 mux. So, normally this N is a power of 2 say 2 to power M, M is integer. So, we require how many 2 by 1 multiplexers are required? To construct N minus 1, 2 by 1 multiplexers are required. See here N is equal to 4, so, we require N minus 1, 3 multiplexers; here N is equal to 8 we require 7 multiplexers.

So, in general so, this N is of course, a power of 2. So, we require N minus 1 2 by 1 multiplexers. So, we can write the Verilog code corresponding to those multiplexers.

(Refer Slide Time: 31:16)



S_1 S_0	Y_0	Y_1	Y_2	Y_3
0 0	I	x	x	x
0 1	x	I	x	x
1 0	x	x	I	x
1 1	x	x	x	I

x_1 x_0	y_0	y_1	y_2	y_3
0 0	E	x	x	x
0 1	x	E	x	x
1 0	x	x	E	x
1 1	x	x	x	E

Now, the next control logic which is the combinational circuit is demultiplexer opposite to this multiplexer is demultiplexer. We have discussed about this decoder, encoder multiplexer, and then opposite operation is demultiplexer. So, this demultiplexer as the name implies this is reverse to that of multiplexer. So, multiplexer will be having 2 raised to the power of n inputs, 1 output and n selection lines whereas, here demultiplexer will be having 1 input and 2 raised to the power of n outputs and n selection lines.

Selection lines are same in both the cases. These are n selection lines. This is called 1 by 2 raised to the power of n demux. If I take a simple example of 1 by 4 demultiplexer, S1 S0 are the selection signals; input is only one I. The outputs we have Y0 if I call this outputs as Y0 Y1 Y2 Y3; if it is 0 0, Y0 becomes I this will be don’t cares. 0 1 this will be don’t care, this is I don’t care, don’t care.

1 0 don’t care, don’t care, this is I only I don’t care and for 1 1 don’t care; don’t care; don’t care only 1 input I ok. So, this input I will be transferred to either Y0 Y1 Y2 or Y3 depends upon the selection signals ok. So, what will be the logic diagram for this one? One interesting thing here is so, this demultiplexer and decoder are almost same.

If I consider a 2 by 4 decoder with enable signal. So, what is the operation here which we have discussed in the earlier classes? This is 2 by 4 decoder, we will be having as the name implies 2 inputs, 4 outputs and 1 enable signal. So, what is the operation? If enable is equal to 1, then the inputs if I call this inputs as x1 x0 these are y0 y1 y2 y3.

If x is equal to 1 what is the operation? x1 x0, output y0 y1 y2 y3, if this is 0 0, y0 is selected. So, y0 will be 1 which is you can call as E. In fact, because E is equal to 1 and these are all don't cares, of course, 0. We don't care about that. So, 0 1 this is don't care E X X. So, don't care is normally 0 we can take now don't care as a 0. 1 0, X X E X, 1 1 X X X E. So, these two table is exactly similar to this.

We can easily obtain a demultiplexer from the decoder by exchanging the inputs and outputs. Here we have 1 input 2 selections 2 raised to the power of n outputs. In general if I take general decoder. So, a general decoder will be having if I write here decoder will be having 10 inputs, 1 enable and 2 raised to the power of n outputs. This is general decoder, this is called n to 2 raised to the power of n decoder.

You see the similarity between these two diagrams. The number of outputs are same here ok 2 raised to the power of n, 2 raised to the power of n. Here one input is there, n selection signals are there whereas, here n inputs are there one selection or enable signal is there ok. So, in order to obtain a demultiplexer from the decoder or vice versa we have to exchange only the inputs and selection lines.

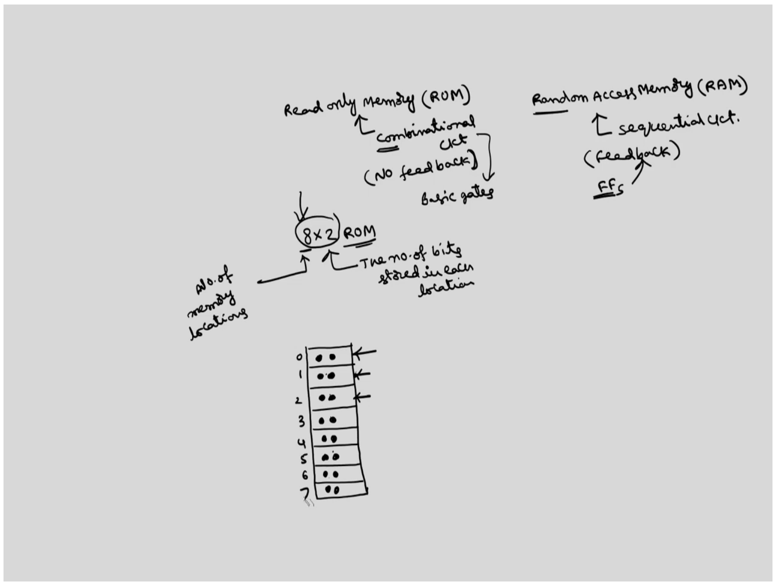
If I transfer this selection lines to here and if this input to selection line then you will get this. Say demultiplexer can be obtained by interchanging the input and selection signals or you can call enable. So, the circuit diagram everything is same. Whatever the circuit that we have discussed for this decoder the same circuit diagram is valid for the demultiplexer also, the only change is this we have to at the place of input we have to apply selection; at the place of selection we have to apply the input.

So, Verilog code is also exactly same, but you have to replace the selection by inputs and inputs by selections. So, this is one of the important feature of this demultiplexer;

demultiplexer and decoder both are almost same except for the change that so, we have to exchange the selections and inputs ok. So, this is another important combinational circuit.

So, we have discussed about this adder, subtractor, then combination of adder subtractor and then we have discussed about this multiplier, then we have discussed about this decoder, encoder, multiplexer, demultiplexer. These are most commonly used combinational circuits.

(Refer Slide Time: 39:06)



So, in addition to that, read only memory is also a combinational circuit called as ROM. There are two types of memory called Read Only Memory and Random Access Memory called RAM. So, this is actually a sequential circuit whereas, read only memory is a combinational circuit.

This is an interesting point. So, what is the difference between sequential and combinational? So, basically the combinational circuit will not have any feedback. So, output it depends upon only the present input without regard to the previous outputs whereas, sequential circuits will be having feedback.

So, if you see the construction of the read only memory. So, this read only memory can be constructed by using simple basic gates such as decoders, AND gates, OR gates whereas, random access memory has to be constructed by using a flip flop; flip flop is the basic circuits, but a flip flop will be having feedback connection.

So, I am going to discuss in the next lectures the flip flops so, where the output is connected to the input. So, you should have some memory whereas, in case of a combinational circuit. So, the combinational circuit does not have any feedback, this can be constructed by using the basic gates without flip flops.

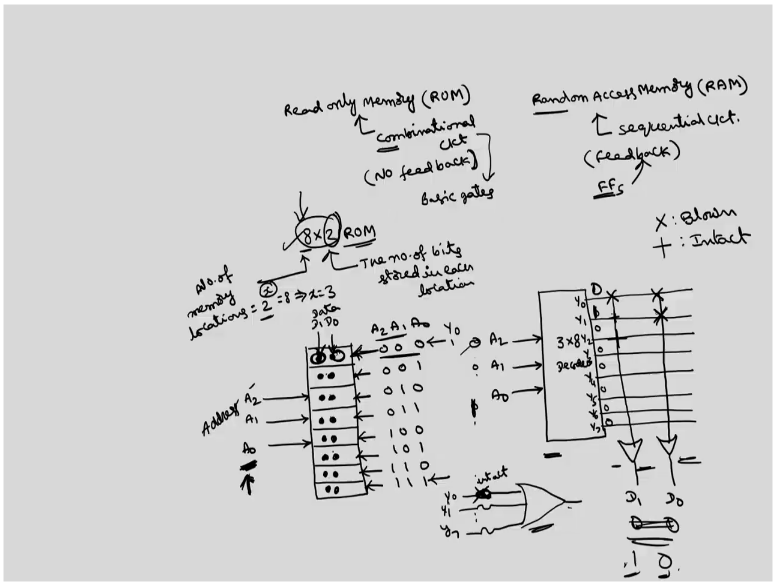
So, if I want to discuss a read only memory, suppose if I want a 8 by 2 read only memory ROM. So, what is the meaning of 8 by 2 read only memory? So, here this 8 by 2 read only memory is the first 8 represent the number of memory locations and this second number represents the number of bits stored in each location; that is this will be having 8 memory locations.

So, 8 locations 1 2 3 4 5 6 7 8 and each location is capable of storing 2 bits of information. Each bit can be either 0 or 1, this can be 0 or 1, this is capable of storing 2 bits of information, this is capable of storing 2 bits of information like that. How many such locations are there? Total we have 8 locations. So, this is 0 location, 1 2 3 4 5 6 7 total 8 locations, that is the meaning of any memory; not only read only memory any memory will be represented by the first digit represents the number of location, the second digit represents the number of bits in each location.

Now, I have 8 different memories. So, to address this how to select this memory? How to select this memory? How to select this memory? For that so, we need some address lines like if you want to find out a house in a particular street so, we have to identify with the house number. So, like that so, this memory also will be having some address lines ok.

So, this address lines for this memory, will be how many address lines are required? So, this in general this first term will be always the number of memory location is power of 2, 2 raised to the power of some x, where x is the number of address lines required if you have x address lines then total I can have 2 raised to the power of x combinations.

(Refer Slide Time: 43:40)



So, here because this is having 8 implies x is equal to 3. So, this 8 by 2 memory will be having 3 address lines A2 A1 A0. The relation between the number of address lines and the number of memory locations is 2 raised to the power of number of address lines is equal to the number of memory locations.

So, this A2 A1 A0 if all are 0 0 0 this location is selected and this is 0 0 1 this location is selected, 0 1 0 this location is selected, 0 1 1 this location, 1 0 0 this location, 1 0 1 this location, 1 1 0 this location, 1 1 1 the last location. So, always this starts with all 0s to all 1s.

So, basically if I take the ROM the internal structure because we have 3 inputs and 8 outputs the first 2 block is 3 to 8 decoder. So, depends upon the number of input this will be number of inputs for the decoder. So, in order to construct any memory, you just see the number of address lines and you give it to the decoder, here if you have n address lines we require n to 2 raised to the power of n decoder here because only 3 address lines are there I require 3 to 8 decoder this is A2 A1 A0.

We know for this combination of this 0 0 0, Y0 is selected; Y0 becomes 1, the remaining all are 0s. So, this will be having 8 outputs. So, this will be having 8 outputs and how to connect this? So, the number of bits in each location this many OR gates you have to use here. I want to store 2 bits in each location. So, I will be having 2 OR gates. If I call this as data, this is address you have called this as address. The contents inside this is called data I will call as D1 D0.

So, this will give D1 this will give D0; D1 is MSB. These are all 8 input OR gates I am not showing 8 inputs. But, I am showing with single line these are all the 8 signals this is nothing, but this Y0 is connected through a fuse Y1 is connected through a fuse like that Y7 is also connected through a fuse to this OR gate these two OR gate that connection we are representing with this. So, here we have all the fuses, this is also another this.

Now, depends upon the data that is to be read or write means read only, that is to be returned at the time of manufacturing we have to blow some connections. If all connections are intact what happens if I give 0 0 0, this will be 1, this will be 0 0 0 0 as we have discussed in the truth table of the decoder. If you have all the fuse intact, then this is 8 input OR gate out of which for all the combinations at least one of the input is 1 always output will be 1 1.

Suppose if I want to store 0 0 in this location 0 0 in this location so, what you have to do? So, whenever if I want to select this location I have to give 0 0 0, this is 0 0 0. So, Y0 is 1 this 1 is 1. So, if I keep all the connections intact output is 1 1, but I want to store 0 0, I want the output as 0 0 what you have to do? You have to blown these two connections this into mark represents blown and this as it is this connection, this connection as it is, this connection represents intact.

So, these two connections if you blow so, for 0 0 0 this is the only 1 output so, but that Y 0 is you have disconnected this one. So, the output become 0 0. In Y1 if you want to store 1 0 what you have to do? So, for 0 0 1, Y 1 is 1 remaining all are 0s. So, if I keep as it is output will be 1 1, but I want this as 1, this as 0. So, this connection I will keep as it is so that this 1 will be transferred.

So, this connection if I keep as it is, 1 will be transferred. So, if I have to intact, we have to blown this. So, like that you can store any data into any location. So, in order to write the Verilog code corresponding to this one so, we can use we can instantiate this 3 to 8 decoder, then we can use the number of OR gates equal to number of data lines and then we can easily write the Verilog code.

So, this is corresponding to some of the important combinational circuits ok. So, in the next lecture we will go for the sequential circuits.

Thank you.