

Lab_HW_#4

M1522.000700 Logic Design (2019 Fall)

2013-12815 이 동 주

1. 3-8 decoder

The image displays two screenshots from the Xilinx ISE Project Navigator. The top screenshot shows the source code for a 3-to-8 decoder module, and the bottom screenshot shows the simulation results.

Source Code (three-to-eight_decoder.v):

```
1 timescale 1ns / 1ps
2
3 module three_to_eight_decoder(
4     input [2:0] A,
5     output [7:0] D
6 );
7
8     input G_L1,
9     input G_L2,
10    input A1,
11    input A2,
12    input B1,
13    input B2,
14    output [3:0] Y_L1,
15    output [3:0] Y_L2
16 //
17
18 );
19 v74x139h_c U1(G_L1(A[2]), A(A[0]), B(A[1]), Y_L1(D[3:0]));
20 v74x139h_c U2(G_L1(~A[2]), A(A[0]), B(A[1]), Y_L2(D[7:4]));
21
22 endmodule
23
```

Simulation Results:

The simulation results show the output of the 3-to-8 decoder. The input A is 000, and the output D is 00000000. The simulation time is 1.000ns.

Object Name	Value
A[2:0]	000
D[7:0]	00000000

WARNING: I5im will run in Lite mode. Please refer to the I5im documentation for more information on the differences between the Lite and the full version.

2. (1) Gate ; Pros – 구현이 쉽다. Cons – 의미 파악이 어렵다.

The screenshot displays a presentation slide titled "Homework" and an ISE Project Navigator window showing Verilog code for a 4-to-1 MUX.

Homework

- Implement 3-to-8 decoder and simulate it
 - You have to re-use your 2-to-4 decoder implemented in lab
- Implement 4-to-1 MUX and simulate it
 - Implement in gate and rtl/behavior level
 - Discuss two methods(Pros & Cons, etc.)
- Implement 16-to-1 MUX and simulate it
 - You have to re-use your 4-to-1 MUX implemented above
- Given a four-input Boolean function $F(A,B,C,D) = \sum m(0,2,4,5,8,10,12,13,14,15)$, implement the function using a 16-to-1 MUX
 - You have to re-use YOUR 16-to-1 MUX implemented above

Verilog Code (four_to_one_mux_gate.v):

```

1 timescale 1ns / 1ps
2
3 module four_to_one_mux_gate(
4     input [0:3] In,
5     input [1:0] C,
6     output Out
7 );
8
9     wire notC1, notC0;
10    wire _In0, _In1, _In2, _In3;
11
12    not U1(notC1, C[1]);
13    not U0(notC0, C[0]);
14    and U3(_In0, notC1, notC0, In[0]);
15    and U4(_In1, notC1, C[0], In[1]);
16    and U5(_In2, C[1], notC0, In[2]);
17    and U6(_In3, C[1], C[0], In[3]);
18    or U7(Out, _In0, _In1, _In2, _In3);
19
20 endmodule
21
  
```

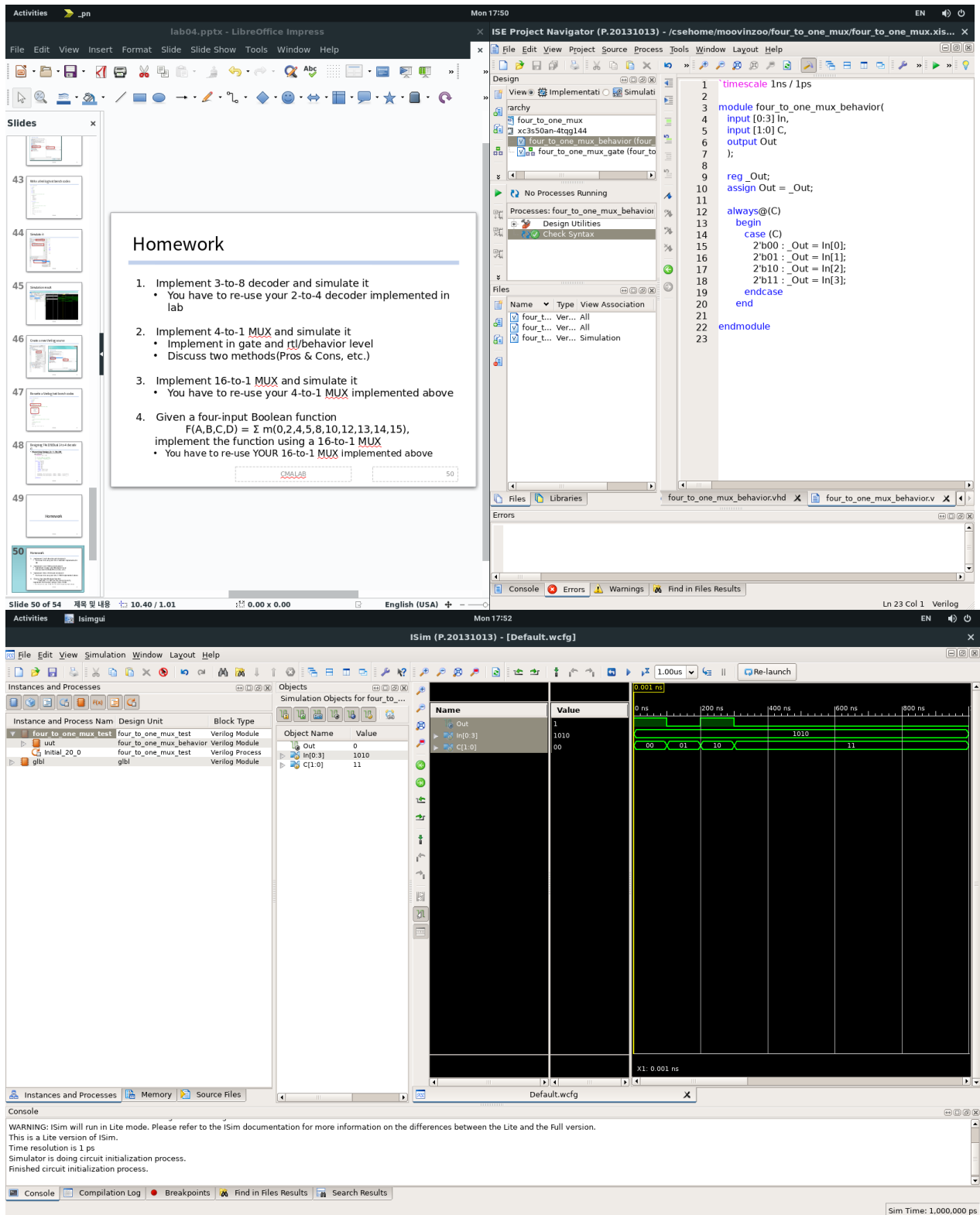
Simulation Results:

The simulation shows the output of the 4-to-1 MUX. The input signals are $In[0:3]$ and $C[1:0]$. The output signal is Out . The simulation time is 1.000s.

Time (ns)	$In[0]$	$In[1]$	$In[2]$	$In[3]$	$C[0]$	$C[1]$	Out
0	0	0	0	0	0	0	0
1010	0	0	0	0	0	1	0
11	0	0	0	0	1	0	0
10	0	0	0	0	1	1	0
11	0	0	0	0	0	1	0

Simulation Time: 1.000,000 ps

2. (2) Behavior; Pros – 논리의 흐름대로 작성할 수 있다. Cons – 구현이 어렵다.



The screenshot displays a computer screen with two main windows. The top window is a LibreOffice Impress presentation titled 'lab04.pptx'. The slide shown is titled 'Homework' and lists four tasks:

- Implement 3-to-8 decoder and simulate it
 - You have to re-use your 2-to-4 decoder implemented in lab
- Implement 4-to-1 MUX and simulate it
 - Implement in gate and rtl/behavior level
 - Discuss two methods(Pros & Cons, etc.)
- Implement 16-to-1 MUX and simulate it
 - You have to re-use your 4-to-1 MUX implemented above
- Given a four-input Boolean function $F(A,B,C,D) = \sum m(0,2,4,5,8,10,12,13,14,15)$, implement the function using a 16-to-1 MUX
 - You have to re-use YOUR 16-to-1 MUX implemented above

The bottom window is the ISE Project Navigator (P.20131013) showing the Verilog code for a four-to-one multiplexer behavior module. The code is as follows:

```

1 timescale 1ns / 1ps
2
3 module four_to_one_mux_behavior(
4     input [0:3] In,
5     input [1:0] C,
6     output Out
7 );
8
9     reg _Out;
10    assign Out = _Out;
11
12    always@(C)
13    begin
14        case (C)
15            2'b00 : _Out = In[0];
16            2'b01 : _Out = In[1];
17            2'b10 : _Out = In[2];
18            2'b11 : _Out = In[3];
19        endcase
20    end
21 endmodule
22
23

```

The ISE Project Navigator also shows the Files pane with the following files:

Name	Type	View	Association
four_to_one_mux	Verilog	All	
four_to_one_mux_behavior	Verilog	All	
four_to_one_mux_gate	Verilog	Simulation	

The bottom window is the ISim (P.20131013) - [Default.wcfg] window, showing the simulation results for the four-to-one multiplexer. The simulation is running at 1.000 ns. The output signal 'Out' is shown as a green trace, and the input signals 'In[0:3]' and 'C[1:0]' are shown as blue traces. The simulation results are as follows:

Time (ns)	In[0]	In[1]	In[2]	In[3]	C[1]	C[0]	Out
0	0	0	0	0	0	0	0
100	0	0	0	0	0	0	0
200	0	0	0	0	0	0	0
300	0	0	0	0	0	0	0
400	0	0	0	0	0	0	0
500	0	0	0	0	0	0	0
600	0	0	0	0	0	0	0
700	0	0	0	0	0	0	0
800	0	0	0	0	0	0	0
900	0	0	0	0	0	0	0
1000	0	0	0	0	0	0	0

The console window shows the following messages:

```

WARNING: ISim will run in Lite mode. Please refer to the ISim documentation for more information on the differences between the Lite and the Full version.
This is a Lite version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.

```

The bottom status bar shows the simulation time: Sim Time: 1,000,000 ps.

3. 16-to-1 MUX by 4-to-1 MUX

The image displays two screenshots of the Xilinx ISE software interface.

Top Screenshot: ISE Project Navigator (P.20131013)

- File:** csehome/moovinzoo/four_to_one_mux/four_to_one_mux.xise - [sixteen_to_one_mux.v]
- Design:** Behavioral view of the project hierarchy. The project is named "four_to_one_mux". The design unit is "four_to_one_mux_test". The design unit contains several components: "xc3s50an-4tag144", "four_to_one_mux_test", "out - four_to_one_mux_gate", "M1 - four_to_one_mux_behavior", "M2 - four_to_one_mux_behavior", "M3 - four_to_one_mux_behavior", "M4 - four_to_one_mux_behavior", and "M5 - four_to_one_mux_gate".
- Processes:** sixteen_to_one_mux_test. The process is running the "ISim Simulator". The process status is "Behavioral Check Syntax".
- Code:** The code is a Verilog module named "sixteen_to_one_mux". It defines a 16-bit input "In", a 3-bit carry input "C", and a 4-bit output "Out". It uses a "timescale 1ns / 1ps". The module contains several behavioral models (M1, M2, M3, M4) and a gate model (M5). The output "Out" is calculated based on the input "In" and the carry input "C".

Bottom Screenshot: ISim (P.20131013) - [Default.wcfg]

- Instances and Processes:** The design unit "four_to_one_mux_test" is shown. The design unit contains several components: "M1", "M2", "M3", "M4", "M5", "Initial_19_0", and "gbl".
- Simulation Object:** The simulation object is "Out". The value is "1".
- Waveform:** The waveform shows the output "Out" over time. The output is a square wave that transitions from 0 to 1 at approximately 1.564481 ns. The input "In" is a constant 1. The carry input "C" is a constant 0.

4. Function Implement

The screenshot displays the ISE Project Navigator (P.20131013) interface. The main window shows the Verilog code for the `sixteen_to_one_mux_function.v` module. The code is as follows:

```
1 timescale 1ns / 1ps
2
3 module sixteen_to_one_mux_function(
4     input [0:15] Bool,
5     input A, B, C, D,
6     output F
7 );
8
9     wire [0:3] outArr;
10
11
12     four_to_one_mux_behavior M1(.In(Bool[0:3]), .C({C, D}), .Out(outArr[0]));
13     four_to_one_mux_behavior M2(.In(Bool[4:7]), .C({C, D}), .Out(outArr[1]));
14     four_to_one_mux_behavior M3(.In(Bool[8:11]), .C({C, D}), .Out(outArr[2]));
15     four_to_one_mux_behavior M4(.In(Bool[12:15]), .C({C, D}), .Out(outArr[3]));
16
17     four_to_one_mux_behavior M5(.In(outArr[0:3]), .C({A, B}), .Out(F));
18
19
20 endmodule
21
```

The left pane shows the project hierarchy, including the `sixteen_to_one_mux_function` module and its testbench. The bottom pane shows the simulation results, including the console output and the waveform viewer.

The console output shows the simulation results:

```
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.

# run 1.50us
```

The waveform viewer shows the simulation results for the `sixteen_to_one_mux_function` module. The waveform displays the input signals `Bool[0:15]`, `A`, `B`, `C`, and `D`, and the output signal `F`. The time scale is 1.50us. The waveform shows that the output `F` is 0 for the first 1.50us, and then it becomes 1010110010101111 at 1.50us.

5. 2*2 bit multiplier (구현을 다 하지 못했습니다)

