Encoders and Decoders

Lab Summary

In this lab, you gain additional experience in gate-level logic design, VHDL, and ModelSim. The two designs incorporated in the lab are commonly used in digital circuitry, the decoder and encoder. These two circuits can be used to change digital data or states from Binary to Hexadecimal or Binary Coded Decimal representation.

Lab Supplies

- None

Lab Videos

- ModelSim
- VHDL Part 1

Part 1: Decoder

If needed, you may wish to review the videos **ModelSim** and **Introduction to VHDL**.

Remember that the address you need to type under EDA Tool Options is:

C:\intelFPGA_lite\18.1\modelsim_ase\win32aloem .

You will design a 2 to 4 Decoder. Begin by constructing a Karnaugh map for each output to find the associated Boolean expressions. From the Boolean expressions, construct the circuit in a new .**bdf** file using the required gate symbols.

Below is the truth table for the 2 to 4 decoder.

Inputs		Outputs			
A1	A0	D3	D2	D1	D0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Figure 1. Truth Table for 2 to 4 Decoder

By this point in the semester you should be well experienced with Karnaugh maps to simplify the circuitry for digital design. Use this technique to derive each of the four equations for the D0, D1, D2, and D3 outputs in terms of the A0 and A1 inputs. You will set up four very simple Karnaugh maps of the form:

A1 ↓ / A0	0	1
0		
1		

You can then construct the decoder schematic (HINT: you should only need NOT gates and AND gates. Of course, you should include your derived equations and the schematic in your report.

Simulate your design using ModelSim. Set your inputs the same way you did for the AND gate in the previous lab, by making two clocks with one input having twice frequency of the other clock.

Checkpoint 1: Show the ModelSim simulation of your 2 to 4 Decoder to your Lab Monitor or TA.

To compare the process, you will next design the same 2 to 4 decoder in VHDL. Start by creating a new VHDL file. Below is the code for the 2 to 4 decoder with the Boolean expressions edited out.

```
1   LIBRARY ieee;
2   USE ieee.std_logic_1164.all;
3
4   ENTITY DECODERVHDL IS
5       PORT
6       (
7           A :   IN STD_LOGIC_VECTOR(1 DOWNTO 0);
8           D :   OUT STD_LOGIC_VECTOR(3 DOWNTO 0)
9       );
10  END DECODERVHDL;
11
12  ARCHITECTURE behavioral OF DECODERVHDL IS
13
14  BEGIN
15
16      D(3) <= [redacted];
17      D(2) <= [redacted];
18      D(1) <= [redacted];
19      D(0) <= [redacted];
20
21  END behavioral;
```

Figure 2. VHDL Code for 2 to 4 Decoder

Some of the expressions you may (or may not) use for your Boolean expressions are: **and, or, not, nor, nand**. For example in VHDL you would write

D(2) <= A(1) and not A(0);

Once you finish typing your code for the 2 to 4 decoder, set the VHDL file as your top-level entity and recompile your project.

Note: You do not have to create a symbol file for the VHDL file if you're only testing it in ModelSim. Also, make sure to name your **VHDL** file for your 2 to 4 decoder with a different name than the one that you used for your 2 to 4 decoder **block diagram** file.

Once, the project is compiled, run the simulation on ModelSim. You should be able to see the same results as when you simulated the block diagram.

Checkpoint 2: Show the VHDL code of your 2 to 4 Decoder and ModelSim simulation to your Lab Monitor or TA.

Part 2: Encoder

Now you will design a 4 to 2 *encoder*, using the same steps you went through to create the 2 to 4 decoder. Include your derived equations and the schematic in your report.

Below is the truth for the 4 to 2 encoder.

Inputs				Outputs	
A3	A2	A1	A0	D1	D0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Figure 3. Truth Table for 4 to 2 Encoder

Any input combination other than the four combinations listed in the truth table will create a “don’t care” output. Place an 'X' in the "don't care" locations, populate and derive equations using the following Karnaugh map. Recall that you can group “don’t care” outputs with ones, in order to reduce the amount of hardware required. (HINT: You should only need two identical gates for this design).

A3 A2 ↓ / A0 A1 →	00	01	10	11
00				
01				
10				
11				

Once you derive the expressions using Karnaugh maps, design the encoder using gates in a new **.bdf** file. Simulate your encoder in ModelSim. For the encoder use clocks with periods of 50 ps, 100 ps, 200ps and 400 ps. With 50 ps being the period of the clock input for the least significant bit. This way you generate all the possible input combinations for a 4-bit input. Run the simulation for at least 1000 ps. Move the yellow cursor all the way to the left, and expand the waveforms with the + magnifying glass until all 4 inputs are visible. You must have at least one complete cycle of all encoder combinations visible when you save the image for your report.

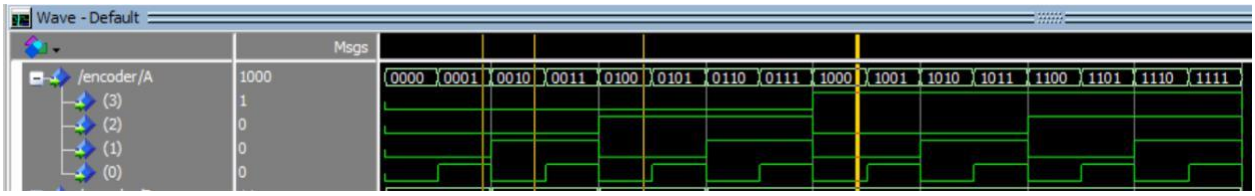

Figure 4. Simulation for 4 to 2 Encoder

Verify that you are getting the right output for all four input combinations.

Checkpoint 3: Show the Block Diagram File and ModelSim simulation of your 4 to 2 encoder to your Lab Monitor or TA.

Next, you will design the same 4 to 2 encoder using VHDL. Refer to your Decoder VHDL code if you need help starting the code. Once you type the VHDL code, simulate it in ModelSim as well.

Checkpoint 4: Show the VHDL code of your 4 to 2 Encoder and ModelSim simulation to your Lab Monitor or TA.

Checkpoint 5: Clean up and show your workbench to the lab monitor or TA.

Discussion Questions

1. If you wanted to design a circuit to monitor the state of 16 discrete digital inputs using the least number of bits on an 8-bit input port, what circuit would you design?

2. If you wanted to control the state of 14 discrete digital outputs using the least number of bits from an 8-bit microcontroller output port, what intermediate circuit would you design? How many output port bits are required? Are there any unused input combinations? If so, how many?

3. The Boolean functions for the outputs of an encoder can be easily derived from looking at the truth table. The outputs are simply an 'or' of the inputs that are '1' when the output is also 1. For example, for the 4 to 2 encoder, D1 = A3 + A2. Using this logic, derive the simplified Boolean expressions for the outputs for an 8 to 3 encoder (aka the Octal to Binary encoder) from the truth table below (without using Karnaugh maps).

Inputs								Outputs		
A7	A6	A5	A4	A3	A2	A1	A0	D2	D1	D0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1