**CST 133**

**Digital Logic II with Verilog**

**Lab 3: Dataflow design of various logic elements**

**Objective: Design structured circuit using dataflow or behavioral modeling techniques.**

# Task 1: Designing 2:4 decoder with tristate outputs

You are to create all of the dataflow modules for a special decoder design and demonstrate it on the FPGA development kit. The block diagrams of the top-level module the various logic designs are shown below:

**2**

**:**

**4**

**Decoder**

**With**

**Tri**

**-**

**state outputs**

a

0

a

1

ena

ble

p

y

0

y

1

y

2

y

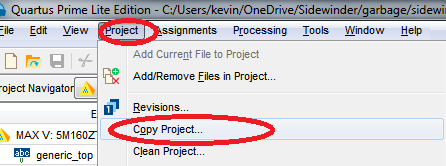
3

The truth table for this special decoder is shown below:

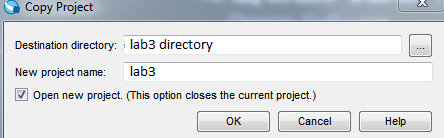
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| enable | p | a1 | a0 | y0 | y1 | y2 | y3 |
| **0** | **0** | **0** | **0** | **1** | **0** | **0** | **0** |
| **0** | **0** | **0** | **1** | **0** | **1** | **0** | **0** |
| **0** | **0** | **1** | **0** | **0** | **0** | **1** | **0** |
| **0** | **0** | **1** | **1** | **0** | **0** | **0** | **1** |
| **0** | **1** | **0** | **0** | **0** | **1** | **1** | **1** |
| **0** | **1** | **0** | **1** | **1** | **0** | **1** | **1** |
| **0** | **1** | **1** | **0** | **1** | **1** | **0** | **1** |
| **0** | **1** | **1** | **1** | **1** | **1** | **1** | **0** |
| **1** | **X** | **X** | **X** | **Z** | **Z** | **Z** | **Z** |

**From the above truth table - Develop the structure for this design and write the Verilog code.**

1. Download the template project provided on the course website.
2. Unzip the generic\_top project
3. Open the generic\_top.qpf file.
4. Go to PROJECT | COPY PROJECT



1. Enter lab3\_decoder as the New Project Name. Make a destination directory called lab3\_decoder. Also enter the new project name as lab3\_decoder.



1. Create a Verilog structural model for a 2:4 decoder within this project. Call it decoder2x4.v
2. The file should start with the following inputs and outputs.

module decoder2x4(

input enable,

input p,

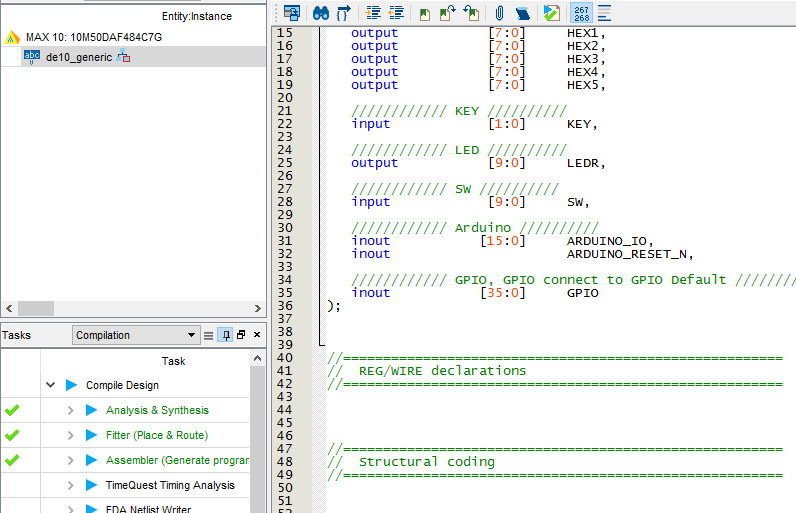
input [1:0] a,

output [3:0] y

);

This module receives four input signals (a0, a1, p, enable) and generates four output signals (y[0:3]). Use the compiler/synthesizer to work out errors in your syntax.

1. Code the truth table for the decoder into the decoder2x4.v file.
2. Double-click on de10\_generic in the project navigator. The de10\_generic.v file should appear. This is your top level file. Before the endmodule, you will instantiate a copy of your decoder2x4.v file.



1. Scroll all the way down to the bottom. Add an instance of your decoder to the de10\_generic.v file before endmodule. You can add an instance of decoder2x4 by doing the following:

decoder2x4 kevinisawesome(.enable(),.p(),.a(),.y()).

1. You’ll want to use the following pin mappings. Notice the empty parenthesis above. Please fill in the appropriate LED(s) and switch(es) for each input and output.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| enable | p | a1 | a0 | y0 | y1 | y2 | y3 |
| SW[3] | SW[2] | SW[1] | SW[0] | LEDR[3] | LEDR[2] | LEDR[1] | LEDR[0] |

1. Program and test the design on the DE10-Lite development board. Refer to instructions in Lab 1, Lab 2, or the Youtube video for how to program.
2. Demo to instructor, clean project, and zip up project files.

# Task 2: 4:2 Encoder with Enable Input and Valid Output

Create the following priority encoder design and demonstrate it on the DE10-Lite development board. The block diagrams of the top-level module the various logic designs are shown below:

**4**

**:**

**2**

**Encoder**

**with**

**Enable input**

**&**

**Valid output**

a

0

a

1

ena

r

0

r

1

Valid

a

2

a

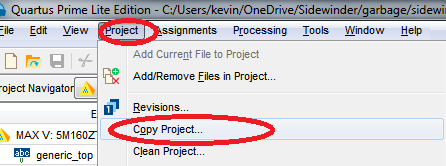
3

The truth table for the encoder is shown below:

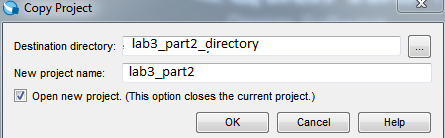
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ena | a3 | a2 | a1 | a0 | r0 | r1 | valid |
| **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** |
| **0** | **0** | **0** | **0** | **1** | **0** | **0** | **0** |
| **0** | **0** | **0** | **1** | **X** | **0** | **1** | **0** |
| **0** | **0** | **1** | **X** | **X** | **1** | **0** | **0** |
| **0** | **1** | **X** | **X** | **X** | **1** | **1** | **0** |
| **1** | **X** | **X** | **X** | **X** | **0** | **0** | **1** |

**From the above truth table - Develop the structure for this design and write the verilog data flow code.**

1. Download the template project provided on the course website.
2. Unzip the de10\_generic project
3. Open the de10\_generic.qpf file.
4. Go to PROJECT | COPY PROJECT



1. Enter lab3\_encoder as the New Project Name. Make a destination directory called lab3\_encoder. Also enter the new project name as lab3\_encoder.



1. Create a Verilog structural model for a 4:2 encoder within this project. Call it encoder2x4.v
2. The file should start with the following inputs and outputs.

module encoder2x4(

input ena,

input [3:0] a,

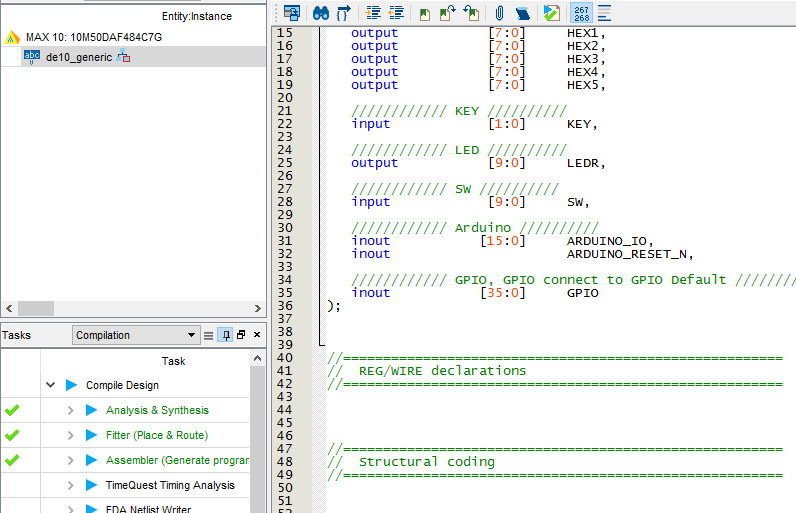
output [1:0] r,

output valid

);

This module receives four input signals (a[3:0], and ena) and generates three output signals (r[1:0] and valid). Use the compiler/synthesizer to work out errors in your syntax. This module receives five input signals (a3, a2, a1, a0, ena) and generates three output signals (r1, r0, valid). Use the compiler/synthesizer check for errors in syntax.

1. Code the truth table for the decoder into the encoder2x4.v file.
2. Double-click on de10\_generic. The de10\_generic.v file should appear. This is your top level file. Before the endmodule, you will instantiate a copy of your encoder2x4.v file.



1. Scroll all the way down to the bottom. Add an instance of your decoder to the de10\_generic.v file before endmodule. You can add an instance of encoder2x4 by doing the following:

encoder4x2 kevinisawesome(.enable(),.a(),.r(),.valid()).

1. You’ll want to use the following pin mappings. Notice the empty parenthesis above. Please fill in the appropriate LED(s) and switch(es) for each input and output.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ena | a3 | a2 | a1 | a0 | r0 | r1 | valid |
| SW[4] | SW[3] | SW[2] | SW[1] | SW[0] | LEDR[2] | LEDR[1] | LEDR[0] |

1. Compile, synthesize and download your design to the DE10-Lite FPGA development board. See lab 1, lab 2, or the Youtube video for instructions.
2. Demonstrate the design to the instructor.

# Submissions

The instructor does not guarantee last second availability. It is your responsibility to submit the lab in a timely manner and to contact the instructor before lab is due. Note that the instructor’s mailbox may not be open after 5 PM.

A penalty of 20% per day up to 4 days will apply. After 4 days, no submissions will be accepted and you will receive a score of 0 for the lab.

For this lab you must submit the following to your instructor’s mailbox or electronically by the date and time shown on the course website.

Submit the following:

1. Zipped up folder for lab3 encoder
2. Zipped up folder for lab3 decoder
3. Demo the decoder before next lab.
4. Demo the encoder before next lab.