Lab Exercise #3 -- Combinational Components

A. The "sim" Software Package

In this course, you will use a software package named "sim" to study circuits. A brief introduction to "sim" is available as:

~cse320/General/intro.sim

If you have not already done so, modify your ".personal" file as follows:

1. Add the following line to your ".personal" file, after any existing "setenv PATH" statements:

setenv PATH {$PATH}:/user/csearch/bin

2. Execute the following command:

<prompt> source .personal

You won't need to repeat these two steps during subsequent sessions.

B. Multiplexers

A multiplexer uses N control signals to select one of 2^N data signals to be mapped to the output signal.

The file "~cse320/Labs/lab03.mux.c" contains a C++ module which is a test bed for experimenting with the multiplexer ("Mux") component.

1. Bring the module into simulation using the UNIX command:

<prompt> sim ~cse320/Labs/lab03.mux.c

a) Give the initial value of all signals which are inputs to the selection and data ports on this 4-to-1 Mux.

C1: _____ C0: _____ D3: _____ D2: _____ D1: _____ D0: _____

b) What is the initial value of the output from the Mux? Explain.

c) Which of the inputs to the data ports can be changed using the switches? Which cannot? Explain.

2. Experiment with the test bed and answer the following questions.

a) Assert both control signals. What is the output from the Mux? Explain.

b) De-assert one control signal, leaving the other signal asserted. What is the output from the Mux? Explain.

c) Reverse the values of the two control signals (one asserted, the other de-asserted). What is the output from the Mux? Explain.

d) Experiment with the data inputs: assert a subset of the data inputs, then show that the Mux routes one of the data inputs to its output, depending on the control inputs. Give the characteristic table (abbreviated truth table) for this multiplexer, and classify it using notation like "N-to-1 Mux".

3. Copy the C++ module into your account, then modify it to allow the user to vary any of the inputs to the data port (by adding another switch).

C. Banks of Multiplexers

In many situations, it is useful to create a bank of multiplexers to choose between multi-bit data signals. For example, a bank of sixteen 4-to-1 Muxes could be used to select one 16-bit data signal out of a set of four 16-bit data signals and map it to the 16-bit output signal.

The file "~cse320/Labs/lab03.mux_bank.c" contains a C++ module which is a test bed for experimenting with a bank of multiplexers.

In that module, function "Mux421" is invoked with three parameters: a schematic descriptor (so that the row and column indicators inside the function are relative positions), an input signal, and an output signal.

1. Examine that C++ source code module and answer the following questions.

a) How should this bank of multiplexers be described? That is, how many data signals are there? How many output signals are there? How wide is each data and output signal?

b) How wide (in bits) is the input signal parameter? What does each subset of bits within that input signal represent?

c) How wide (in bits) is the output signal parameter? Explain.

2. Bring the module into simulation using the UNIX command:

```
<prompt> sim ~cse320/Labs/lab03.mux_bank.c
```

Note: the 'Page Down' key with the 'F12' key to redraw) can be used to examine the inside of a module; the 'Page Up' key can be used to back up.

a) Use the keys 'a' to 'd' to set the Port 1 data signal to a particular bit pattern. What output is displayed on the output probes? Explain.

b) Uses the keys '1' to '4' to set the Port 0 data signal to a particular bit pattern. Now, what output is displayed? Explain.

3. The "Mux" component in "sim" supports N-bit data and output signals directly: the width of the output signal determines the width of each of the data signals. Use "simhelp Mux" to view the brief component overview.

Copy the C++ module into your account, then modify it to use a standard "Mux" component (with 4-bit data and output signals) instead of function "Mux421".

D. Decoders

The file "~cse320/Labs/lab03.decoder.c" contains a C++ module which is a test bed for experimenting with the decoder components.

1. Bring the module into simulation using the UNIX command:

```
<prompt> sim ~cse320/Labs/lab03.decoder.c
```

a) Initially, all of the probes display the value "Zero", indicating that none of the decoder outputs are asserted. Explain.

b) Touch the 'e' key to enable the decoder. What do the probes display? Explain (the 'F8' function key may be helpful).

c) Touch the ’r’ key to reset the counter. What values do the probes display? Explain.

d) Touch the ’s’ key three times to make the counter move through the sequence {0, 1, 2, 3}. What values do the probes display? Explain.

e) Touch the ’r’ key to reset the counter. What values do the probes display? Explain.

f) Touch the ’e’ key to disable the decoder. What values do the probes display? Explain.

2. Copy the C++ module into your account, then modify it so that it uses a 3-bit counter and a 4-to-16 decoder. The most significant bit of the data input to the decoder will be controlled by a switch.