CALIFORNIA STATE UNIVERSITY
LOS ANGELES

Department of Electrical and Computer Engineering

EE-2449 Digital Logic Lab

EXPERIMENT 7

MULTIPLEXERS AND DECODERS

Text: Mano and Ciletti, *Digital Design, 5th Edition,* Chapters 3 and 4

Required chips: **74155**: Decoder. **7410:** triple 3-input NAND**. 74151:** 8x1 MUX.

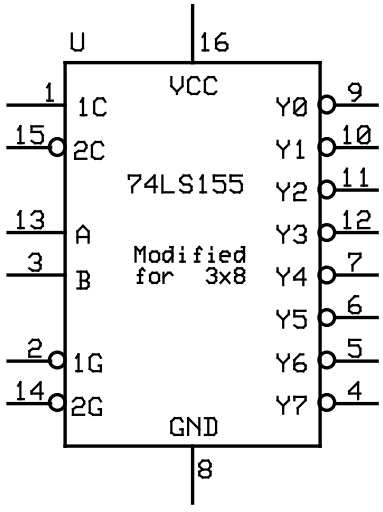
This experiment deals with alternative ways to build a combinational circuit (one without storage elements--flipflops and registers).

In 7.1, your design (which you will build and test) uses a decoder and a couple of gates. This is a much simpler design approach because most of the gates involved are built in to the decoder.

In 7.2, your design (again, to be built and tested) is based on a multiplexer (MUX). No external gates are required, so this is probably the simplest design in the experiment.

7.1* Design with Decoders: (See Decoders, Ch.4 of Mano and Ciletti). Connect the 74LS155 as a 3x8 decoder. Its characteristics are discussed at the end of this manual in the section called Descriptions of ICs.

The diagram at the right is a modified version of the one found in the manual and is better suited to this experiment. Pin names are the same, but pins at left and right are arranged in a more convenient manner. This symbol, like all custom symbols, is available from the Component and Symbol Manager under *Library Components* in lab or *Custom Components*.in your own computer (i.e. in your ExpressPCB folder under My Documents). Use this symbol **only**.



Decoders generate minterms. A 3x8 decoder will generate the 8 minterms for the 3 inputs. A decoder with built with NAND gates will have active-low outputs, and thus, will generate maxterms instead of minterms. The 74LS155 has active-low outputs as indicated by the bubbles on the output pins Y0 through Y7.

Any function can be built as a sum of minterms or a product of maxterms.

Question 7.1.1: If a decoder with active-high outputs generates minterms, what type of gate can be used with the decoder to implement functions that are expressed as sum of minterms?

Question 7.1.2: Likewise, what type of gate can be used with a decoder with active-low outputs to generate a product of maxterms?

Question 7.1.3: How can an AND gate be built from NAND gates? Draw your solution in your lab notebook.

Question 7.1.4: Draw the two representations for a NAND gate in your lab notebook – AND-NOT and NOT-OR.

Choose functions F1(X,Y,Z) and F2(X,Y,Z) using the approach of Exp. 6.2 (i.e. ***draw their maps***).

- F1 should have the form "xxx + xx"; example: XYZ + Y'Z'.
- F2 should have the form "x + xx"; example: X' + YZ'.

Make up your own functions--don't borrow functions from others in class or use the examples given here. Also, as in Experiment 6.2, make sure your functions cannot be further simplified: the map for F1 should have *three* 1's and the one for F2 should have *five*.

Express function F1 as:

- A sum of minterms
- A product of maxterms
- A minimized sum of products

Express function F2 as:

- A sum of minterms
- A product of maxterms
- A minimized sum of products

Implement two functions F1(X,Y,Z) and F2(X,Y,Z) using the 74LS155 decoder chip.

- Connect X, Y, and Z to counter outputs QC, QB, and QA.
- Connect output QC (X) to *both* pins 1 and 15.
- Connect QB (Y) and QA (Z) to pins 3 and 13, in that order.
- Connect pins 2 and 14 to ground. They are active-low enable inputs and must both be low for the decoder to function.

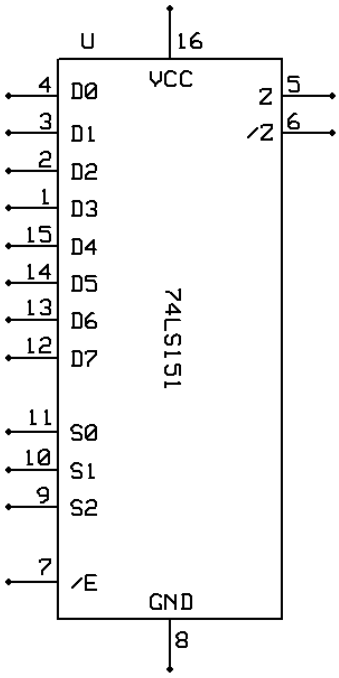
In addition to the decoder, you will need <u>one</u> 7410 triple 3-input NAND chip (nothing else). F1 should be straightforward. F2 may appear difficult or impossible (since only 3-input gates are available), but it's actually easy--it just requires a different approach. (Hint: consider your answers to the questions 7.1.1 through 7.1.4 and also consider the various ways you can express functions F1 and F2).

Testing your design: As in Exp.6, bring X and F1 to the scope. Remember to trigger on the falling edge of X, as before. Adjust Time/Div on the scope so that *one cycle of X spans 8 divisions on the scope screen, one division per count.* Then repeat for X and F2. Include scope images in your report, properly labeled.

Make sure your maps agree with the scope images. If they do, comment on this in your report. If not, check to see if you made a mistake in mapping your function F1 or F2. If the map is OK, you may have a connection error in your circuit, which needs to be fixed *or this section is not complete*.

Question: Which is a more efficient design in terms of number of transistors/gates required: 1) using a decoder and NAND gates, or 2) using discrete gates to implement the minimum sum of products expressions for F1 and F2? Explain your answer.

7.2* Design with MUX's: Normally, a multiplexer or "MUX" is used as a data-selector; its output is selected from among one or more sets of data inputs. However, MUX's can also implement arbitrary boolean functions. They are extensively used for this purpose in programmable chips like FPGA's (field-programmable gate arrays).



The 74LS151 is an 8x1 MUX with 8 data inputs D0-D7, 3 select inputs S0-S2, and one output Z (plus its complement). It also has an active-low enable input /E which must be connected to ground.

Its data input values, 1's and 0's, are taken from the output column of a function's truth-table. All data inputs equal to 1 are to be connected to power (5V); otherwise to ground. To find power and ground symbols for your ExpressSCH drawing, click on Component Manager and open *Library Symbols*.

For a function F(XYZ), connect X,Y,Z to S2,S1,S0. Each value of XYZ (000...111) selects a data input that is sent to output Z. In a sense, the MUX is like the physical counterpart of a truth-table. The value of XYZ corresponds to a row on the left-hand side of a truth-table, and the right-hand side of that row is the selected value of F. So, in this case, a MUX acts like a Look-Up-Table. To look up the value of F for a given select combination XYZ, you apply that combination to the MUX and the answer is supplied at output Z. The term Look-Up-Table is usually shortened to "LUT" in material dealing with FPGA's.

> In FPGA's the LUT's output values can come from an on-chip RAM. By downloading a different set of output values to the RAM while the circuit is in operation ("on-the-fly" as they say), the circuit's function can be completely changed.

IN LAB: Use the 74LS151 to implement a function F(X,Y,Z). Choose a function with the form **F(X,Y,Z) = xxx + xx + xx**, (where x represents an input or its complement).

Proceed as follows: (1) map the function, (2) draw the circuit using the custom symbol for the 74151, (3) connect the MUX inputs D0-D7 according to the map (1's connect to 5V, 0's connect to ground), and (4) verify circuit operation in the usual way. Let XYZ come from counter outputs QC, QB, and QA, where the counter is triggered by a fast clock. Display F vs. X on the 'scope. Trigger the scope display on the falling edge of X. Show a full cycle of X with falling edge at the left. Make sure the scope image agrees with your truth-table. If not, *fix your connections so it does*.

SUMBMIT A LAB REPORT FOR EXPERIMENTS 6 AND 7 COMBINED.