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

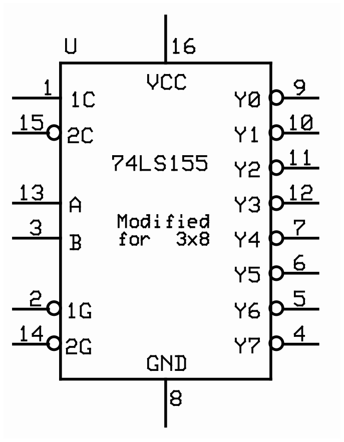
Required chips: **7493**: Counter. **74155**: Decoder. **7410:** triple 3-input NAND**. 74151:** 8x1 MUX.

This experiment deals with alternative ways to build a combinational circuit (one without storage elements; i.e. no flipflops or registers).

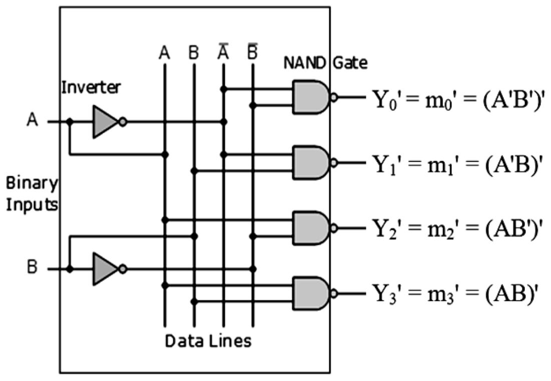
In 7.1, your design (which you will build and test) uses a decoder and a couple of gates. This is a much simpler design approach because most of the gates involved are built in to the decoder.

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**.



At the right is a 2x4 decoder, a smaller version of the 74155 showing the connections inside. The diagram was obtained online and the labels used are different from the 74155. In the 74155, C is the msb (most significant bit) so X connects to C. In the 2x4 diagram, A is the msb so X connects to A (not B). Otherwise, the basic construction is the same.



As you can see, the decoder outputs all possible minterms (complemented). So by including a decoder in your circuit, you don't have to use gates to *create* the minterms that you want (as in Exp 6), you only have to *select* them at the decoder output and then collect them together with an external OR gate, or an invert-OR in this case.

An invert-OR has bubbled inputs which cancel out the bubbles at the decoder outputs. (This cancellation of inverter bubbles was discussed in Exp 6.)

In Exp. 6 you started with a map and derived an equation for F by combining squares in the map to simplify the algebra and thereby minimize the amount of hardware (gates) needed for your design. But here, most of the hardware is already inside the decoder so there's no point in combining map squares and simplifying the algebra to save gates. Trying to do that is a waste of time. As it says above, just collect together the individual minterms you need at the decoder output. Using a decoder makes designing much simpler--almost trivial.

Now, 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. The map for F1 should have *three* 1's and the one for F2 should have *five*.

Express your F1 and F1' as sums of minterms. Example: for F1 given above we get Σ(m0,m4,m7) from the 1's of its map and for F1' we get Σ(m1,m2,m3,m5,m6).

Repeat for F2 and F2'.

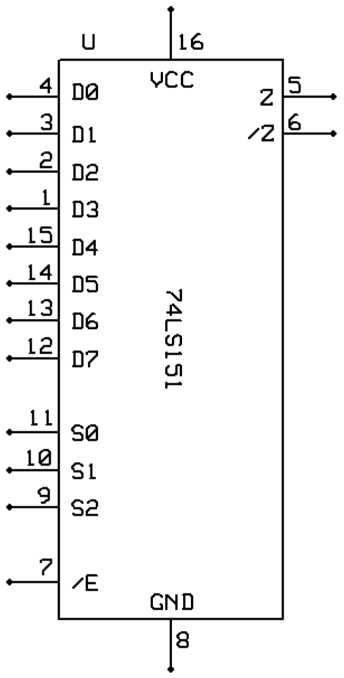
Now, implement the two functions F1(X,Y,Z) and F2(X,Y,Z) using the 74LS155 decoder chip.
- Connect X, Y, and Z to counter outputs QC, QB, and QA.
- Connect X to *both* pins 1 and 15.
- Connect Y and Z to pins 3 and 13, in that order.
- Connect pins 2 and 14 to ground. They are active-low enable inputs and must both be low for the decoder to function.

In addition to the decoder, you will need one 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 (hint: design for F2' and then do……what?).

Testing your design: As in Exp.6, bring X and F1 to the scope, using a fast clock. 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. (Copy screen image to flash drive or use camera.)

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.* (See the Appendix at the end of Exp.4 for what may be an efficient way to troubleshoot your circuit by mapping scope images.)

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, three select inputs S2-S0 (pins 9,10,11), 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 select inputs 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, the circuit's function can be completely changed ("on-the-fly" as they say).

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 and derive its truth table from the map.

(2) Draw the circuit using the ExpressSCH symbol for the 74151. Connect the MUX inputs D0-D7 according to the table (1's connect to 5V, 0's connect to ground). Let XYZ come from counter outputs QC, QB, and QA and connect to them to S2,S1,S0. No need to include the 7493 counter in your diagram; just label inputs as X(QC), Y(QB), and Z(QA)

(3) Connect the circuit as shown in the diagram. Trigger the counter with a fast clock from the timer output (i.e. switch the 0.47uF capacitor into the timer circuit).

(4) Verify circuit operation using the oscilloscope. 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*.

Include scope images in your report, properly labeled.

SUMBMIT A LAB REPORT FOR EXPERIMENTS 6 AND 7 COMBINED.