

## Combinational Logic with Multiplexers and Decoders: Design of a Full Adder

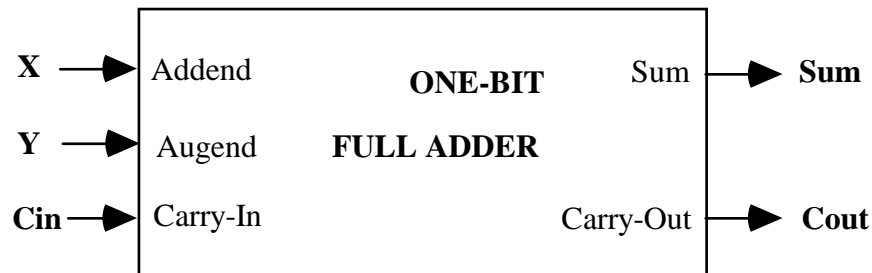
### Part MG3 (First Week)

#### I. Introduction.

In the previous experiments we have investigated the use of three functionally complete sets of gate operations for realization of combinational logic functions: 1) AND, OR, and NOT gates, 2) NAND gates, 3) NOR gates. We have also seen that the XOR gate, while not functionally complete, is often useful for certain applications such as parity function generation. You have used SSI (Small-Scale Integration) IC packages in your laboratory kit of parts to implement these designs.

In this experiment we apply two MSI (Medium-Scale Integration) ICs, the multiplexer and the decoder, to design the combinational logic circuitry of a one-bit binary full adder.

As shown in the diagram below, a one-bit full adder has three one-bit inputs (addend X, augend Y, and carry-in C), and two one-bit outputs (carry-out Cout, and Sum). An N-bit ripple-carry full adder would be obtained by cascading N of these one-bit full adders.



A possible realization of a 1-bit full adder is shown in Figure 3-27 at page 127 of your textbook.

1. Write a truth table for the one-bit full adder, with columns in the order X, Y, C, Cout, Sum. The input values for the eight rows should be in a binary counting sequence.
2. Write Sum and Cout as a **sum of minterms**, both as Boolean expressions and in the shorthand decimal notation. Note that bit C is the least significant bit (LSB), and X is the most significant bit (MSB).

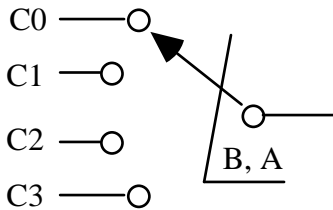
#### II. One-Bit Full Adder Implementation with Multiplexers

A multiplexer is the IC counterpart of a selector switch whose single output can be connected to any one of N different inputs. The "position" of the switch is determined by an input binary number or address having  $\log_2(N)$  bits. An IC package containing several multiplexers with a common address control is the counterpart of a multiple-pole, multiple-throw switch. Examples in the Low-Power Schottky technology are: 74LS157 Quad 2-to-1, 74LS153 Dual 4-

to-1, and 74LS151 Single 8-to-1. The 74LS157 and 74LS153 are included in your laboratory hardware kit. The 74LS153 will be used in this experiment.

The functional equivalent of one section of this 4 to 1 multiplexer is shown here, where the two switch address bits B and A control the position of the switch, so that the output Z is connected to exactly one of the four inputs C0 through C3.

$$Z = (\bar{B} \cdot \bar{A}) \cdot C_0 + (\bar{B} \cdot A) \cdot C_1 + (B \cdot \bar{A}) \cdot C_2 + (B \cdot A) \cdot C_3 \quad [1]$$



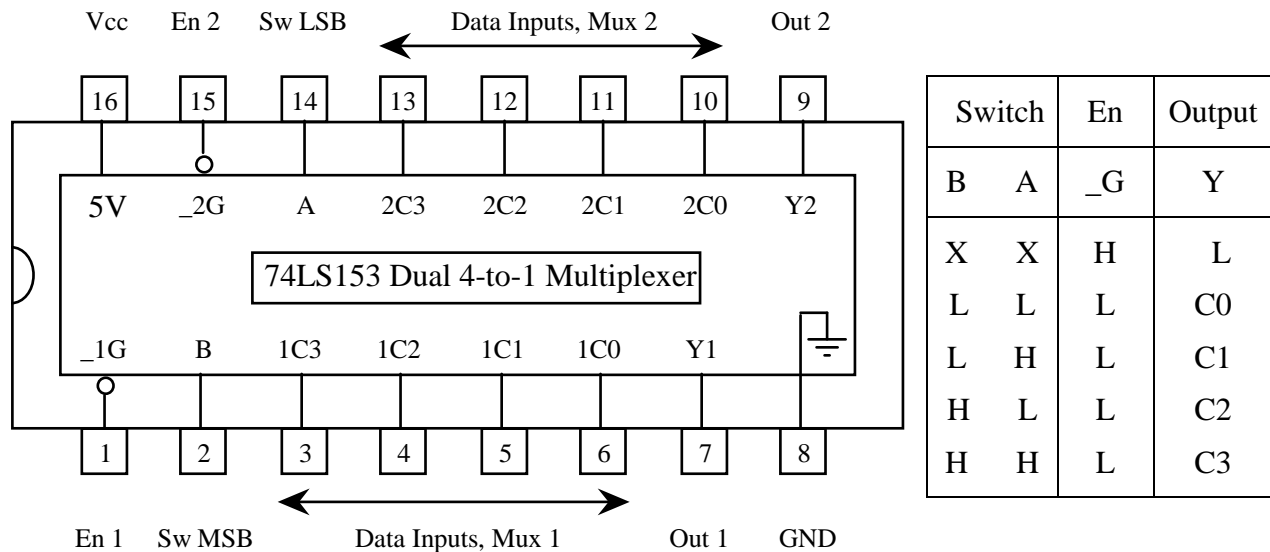
Although the multiplexer is most often used as a switch, it is also used to generate combinational logic. A multiplexer with M address lines can directly realize any function of those M variables because there is exactly one input corresponding to each minterm, and these inputs can be individually set to 0 or to 1 in accordance with the truth table of the function. The multiplexer and one inverter can realize any function of M+1 variables, where the additional variable, or its complement, or 0 or 1 is connected to each input line, as determined from the particular function.

For example, suppose that we have a function of three variables, Z(X, Y, C), which we wish to realize with a 4 to 1 multiplexer. If we use variables X and Y as the two switch address bits for the multiplexer, then C, becomes the "extra" variable, and the four switch inputs C0 through C3 can be evaluated from Equation [1] by substitution of specific values for X and Y as follows:

$$C_0 = Z(0, 0, C) \quad C_1 = Z(0, 1, C) \quad C_2 = Z(1, 0, C) \quad C_3 = Z(1, 1, C) \quad [2]$$

These calculations yield one of the values 0, 1, C, or C' for each expression. The four Ci values may alternatively be determined directly by inspection of the truth table for Z(X, Y, C).

The voltage-level table and connection diagram for the 74LS153 are shown below. Note that B is the *most* significant bit of the two-bit address, and that the enable signal *\_G* is **active low**.



1. Design a circuit to implement the full adder using only this dual 4-to-1 multiplexer and one inverter. Use your logical equations from Part I.2 with the evaluation procedure given by Equation [2] in Part II. Confirm your results by inspection of the truth tables.

### III. One-Bit Full Adder Implementation with a Decoder

A decoder is a MSI combinational logic IC which has  $N$  inputs and  $2^N$  outputs, corresponding to the minterms of the input, such that exactly one output is active for each combination of the input variables. Any function of  $N$  variables can then be obtained by ORing together the appropriate combination of minterms from the output of the decoder. Examples of decoders in the Low-Power Schottky technology are: 74LS155A dual 2-to-4, 74LS138 single 3-to-8, and the 74LS42 single 4-to-10 (BCD to decimal). Since the full adder has three inputs, we will use the 74LS138 single 3-to-8 decoder for this experiment. The pinout diagram and the voltage-level table for this IC are shown on the next page. Note that the outputs are **active low**, as denoted by the circles. Also note that three enable inputs are configured so that enablement requires  $G1$  to be High and both  $\_G2A$  and  $\_G2B$  to be Low.

If  $G1$  is used as a signal input, then the IC is called a demultiplexer, where the complement of  $G1$  is directed to the specific  $Y$  output line selected by the CBA address. The switch analogy would be the 4-position switch on Page 2 with signal propagation from right to left.

1. From inspection of your truth table for the full adder, determine the appropriate connections for implementation with one 3-to-8 decoder, assuming that the decoder outputs are active high, and show a circuit with Sum and Cout as the outputs of OR gates.
2. Since the outputs of the 74LS138 are active low, show that you can modify your circuit of Part 1 to employ this decoder with two NAND gates to realize the full adder. The 74LS20 is a dual 4-input NAND, so this design requires only one 74LS138 and one 74LS20.

### IV. Schematic Creation with Design Architect

Create one schematic diagram which contains both your multiplexer implementation as well as your decoder solution for the one-bit full adder. Three input ports will provide the  $X$ ,  $Y$  and  $C_{in}$  inputs for both designs. Each design will have its own pair of output ports.

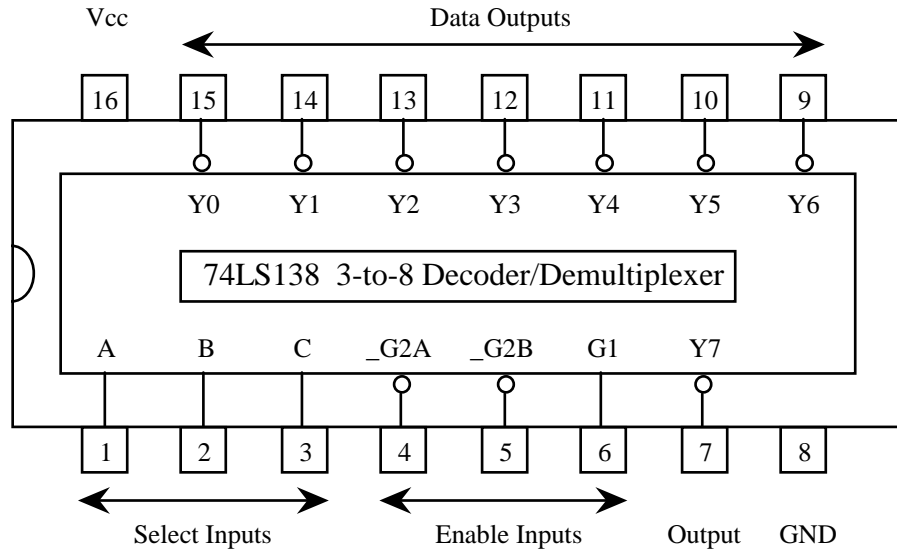
1. Create directory named lab3 under your course directory.
2. Open the design manager and create new sheet with **MG3** for the Component Name in the dialog window.
3. Obtain the following circuit components from the Board Processing Library through the *Library* Button in the palette.

*ground*, *portin*, *portout*, *vcc*, 74LS04 (Inverter), 74LS138 (3-to-8 Decoder),  
74LS153 (Dual 4-to1 Multiplexer), 74LS20 (Dual 4-input NAND)

Obtain the following bus rippers through the pull-down menu:

*Libraries* > *MGC Generic Library* > (*Palette*) > *Rippers/Misc* > *ripper 4x1*, *ripper 8x1*

4. Make copies of the elements as appropriate. Connect all of the enable inputs of the



Enable Inputs		Select Inputs			Outputs							
G1	_G2*	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	H	L	H	H	H	H	H	L	H	H
H	L	H	H	H	H	H	H	H	H	H	H	L

$$_G2^* = _G2A + _G2B$$

multiplexer and decoder to *ground* or *vcc*, as appropriate, so that the multiplexer and decoder are always enabled.

- Use the [ADD/ROUTE] palette commands FLIP, ROTATE and MOVE, as appropriate, to attach 4x1 bus rippers on the inputs of each 74LS20 NAND gate, and the 8x1 ripper on outputs 0-7 of the 74LS138 Decoder. If you place the ripper pins on top of the component pins, yellow circles may appear at each overlay. You can then click the icon CONNECT ALL to make the connections. Use ADD BUS/BUNDLE to perform the bus wiring from the 8x1 ripper to each of the 4x1 rippers.
- Assign the names **X**, **Y** and **Cin** to the input ports, and the name **OUT(7:0)** to the decoder

output bus.

7. Use ADD WIRE to route individual wires to complete your circuit.
8. Assign names to all of the output ports: **Sum\_Mux**, **Cout\_Mux**, **Sum\_Dcd** and **Cout\_Dcd** for the multiplexer and decoder respectively. Use [TEXT] SEQUENCE TEXT to assign appropriate numbers to the ripper R (Rule) values for the address inputs to the multiplexer and the decoder, and the decoder output. Use the auto sequence mode and drag the LMB to form a rectangle enclosing the group to be renumbered. Sequentially enter values for the highlighted items.
9. Use [TEXT] CHANGE VALUE to change the R values at the inputs of the NAND gates to correspond with your decoder design. Drag the LMB to form a rectangle enclosing the group to be renumbered. Sequentially enter your values for the highlighted items.
10. Check the completed schematic sheet. Save the completed schematic.
11. Obtain a printed copy of the schematic.

## V. Simulation with QuickSim II

1. Invoke QuickSimII.
2. Click on OPEN SHEET in the QuickSim Palette to display the schematic diagram.
3. Create a waveform window for the input bus and the four output ports. Place the traces in the following order, from top to bottom: *X*, *Y*, *Cin*, *Cout\_Mux*, *Cout\_Dcd*, *Sum\_Mux*, *Sum\_Dcd*.
4. A stimulus pattern for the input signals has been stored for this problem. In order to use it, your input port names must be X, Y and Cin. Use the pop-up menu to access

*forces* > *from file*, and in the resulting dialog window enter

**\$MGC\_HOME/user/logic/MG3\_forces\_800**      Click *OK*

5. Run the simulation for 800 nanoseconds. Check that the results are correct. If not, you may modify your schematic in design architect, save it, return to Quicksim and use [DESIGN CHG] RELOAD MODEL > *All* to try again.
6. To prepare printed copy of your simulation output, use the pop-up menu in the trace window to select *Setup* > *Window*. In the resulting dialog window, enter the following:

<i>Domain label interval</i> <b>100</b>	<i>Curve height</i> <b>40</b>	<i>Display Radix</i>
<i>Domain pixels interval</i> <b>100</b>	<i>Curve spacing</i> <b>40</b>	<b>Binary</b>
<i>Domain area height</i> <b>40</b>	<i>Margin</i> <b>10</b>	<b>Busses Only</b>
<i>Name area width</i> <b>100</b>	Click <i>OK</i> .	

Print the waveform with *Begin domain* **0**      *End domain* **830**.

7. Click on RESET in the palette, and select only *State* in the dialog boxes. Click *OK*. From the pull-down menu select *Setup* > *Kernel* > *Analysis*, click *Delay* for timing mode in the dialog window, and click *OK*. Run the simulation and obtain a

plot as in Parts 5 and 6.

8. Use palette [DBG GATES] DELTAS to measure and tabulate the delay times in nanoseconds associated with the responses to the input at each of the times 300 and 400 nanoseconds for each of the outputs **Cout\_Mux**, **Sum\_Mux**, **Cout\_Dcd** and **Sum\_Dcd**. Write these eight delay values directly on your waveform plot from Part 7 of this procedure, placing each value beside its associated transition on the output waveforms.

## VI. Design of a Four-bit Adder/Subtractor

In this section you are to construct an unsigned 1-bit full adder/subtractor using a 1-bit full adder and one two-input XOR gate. The adder/subtractor has a fourth input, SUB. When SUB is logical 1, the unit subtracts, and when SUB is logical 0, it adds. The objective here is to create a four-bit adder/subtractor by concatenating four one-bit signed full adder/subtractors, as shown on the next page. In the following steps, you will modify your MG3 schematic to produce a signed 1-bit adder/subtractor by adding one input port and one 74LS86A XOR gate to the 74LS153 dual 4 to 1 multiplexer circuit, save this schematic as MG3\_FAS, create a symbol for MG3\_FAS, instantiate it four times in a new schematic MG3\_FAS4, complete the 4-bit schematic, and simulate its operation.

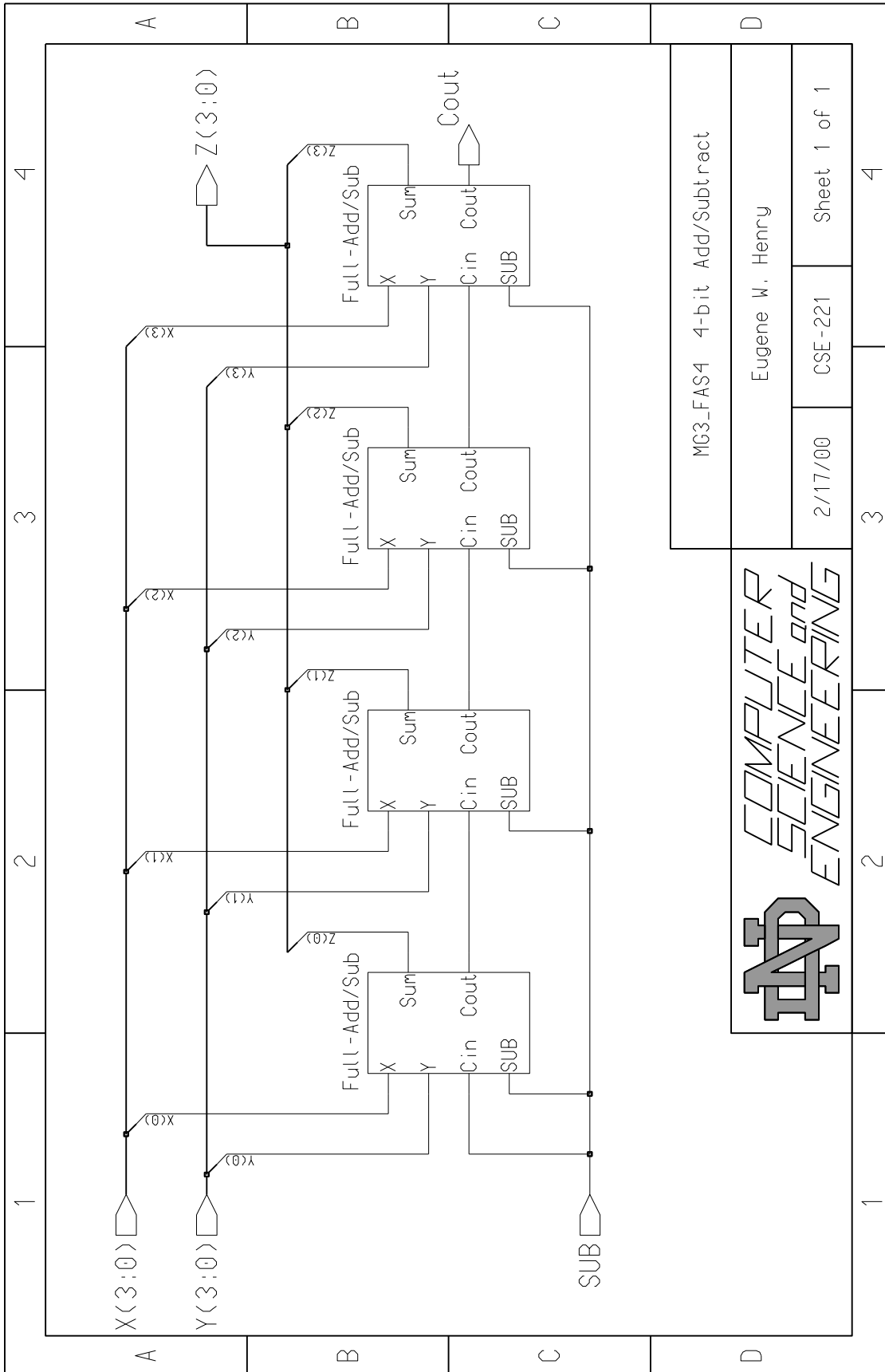
1. With your MG3 schematic in design architect, select and delete all of the components associated with the decoder implementation, i.e. the 74LS138, 74LS20s, etc.
2. Select and copy one of the input ports and rename it *SUB*. Rename the remaining two output ports simply as *Sum* and *Cout*.
3. Add one 74LS86A XOR gate for use with the input SUB. Wire the XOR gate to invert the Y input when SUB = 1. This forms the ones complement of Y. The twos complement is formed by using SUB as the input to Cin(0) in the top level 4-bit schematic.
4. Delete the b-size border and replace it with the smaller a-size border. Check the schematic, and use pull-down *File > Save Sheet As ...* with the name **MG3\_FAS**. Obtain a printed copy of the schematic.
5. Use pull-down *Miscellaneous > Generate Symbol* for the component MG3\_FAS with default options. Edit the symbol so that it appears as shown on the next page. Check and save the symbol.
6. Close the symbol and schematic windows, and open a new schematic called **MG3\_FAS4**. Add an a-size ND border with full-size components, and use the palette *Choose Symbol* to add four instances of MG3\_FAS as shown on page 7. Place three input ports, two output ports and a ground symbol as shown, and rename the five ports as indicated.
7. Use palette ADD BUS/BUNDLE to attach bus lines to the X, Y, and Z ports, as shown. Then use ADD WIRE to complete the diagram. When connecting a wire between a bus and a component input or output pin, always start drawing the wire at the component pin, and terminate at the bus. A ripper will automatically be added at the bus, and a dialog box will request the bit number for that connection.
8. Check and save the MG3\_FAS4 schematic. Obtain a printed copy. Iconify da.
9. Use QuickSim on MG3\_FAS4. Load \$MGC\_HOME/user/logic/MG3\_FAS4\_forces\_800,

and run for 800 ns to print out traces of X, Y, SUB, Cout and Z in signed format.

10. Quit QuickSim without saving, and quit Design Architect.

### **Report**

Your written report should contain the truth table from I.1, minterm expressions from I.2, multiplexer derivation from II.1, decoder derivations from III.1 and III.2, schematics from IV.12, VI.4, VI.8, simulation results from V.6, V.7, VI.9, and discussion of the procedure and results.



### Part HD3 (Second Week)

The purpose of this hardware laboratory is to experimentally verify the operation of the multiplexer and decoder solutions to the one-bit full adder which were simulated in Part MG3 of this experiment. Use your Design Architect schematic as a wiring guide, and prepare a pictorial wiring diagram using the IC package symbols shown on pages 2, 4 and 6 of this experiment.

1. Insert the following four ICs into the breadboard:  
**74LS153** dual 4-to-1 multiplexer, **74LS138** 1-of-8 decoder, **74LS04** hex inverter, and the **74LS20** dual 4-input NAND.
2. Sketch in your lab notebook the **pictorial wiring diagram** for your solutions to the full adder from Part MG3 of this experiment. Use IDL-800 switches S2, S1 and S0 as the inputs **X**, **Y** and **Cin**, respectively, for both circuits. Use the four LEDs L3, L2, L1 and L0 to display the outputs in this order: **Cout\_Mux**, **Cout\_Dcd**, **Sum\_Mux**, **Sum\_Dcd**.
3. Wire the circuits in accordance with your diagram of Part 2.
4. Experimentally determine and record in your notebook the truth table for the four outputs as functions of the inputs from switches S2, S1 and S0.
5. What are the relative advantages and disadvantages of the two solution of this problem? How many 74LS... IC packages are required for each of the two solutions?

