1. (MK 3-23) Construct a 10-to-1 line multiplexer with three 4-to-1 line multiplexers. The
multiplexers should be interconnected and inputs labeled so that the selection codes 0000
through 1001 can be directly applied to the multiplexer selections inputs without added logic.

**10:1 mux**

![10:1 mux diagram]

**Implementation using 4:1 muxes.**

![Implementation diagram]
2. (MK 3-27) Implement a binary full adder with a dual 4-to-1 line multiplexer and a single inverter.

<table>
<thead>
<tr>
<th>A B</th>
<th>C_i</th>
<th>S</th>
<th>C_o</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>00</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>01</td>
<td>0</td>
<td>C_i</td>
<td>0</td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>C_i</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
3. (MK 3-34) Design a combinational circuit that forms the 2-bit binary sum  $S_1S_0$ of two 2-bit numbers $A_1A_0$ and $B_1B_0$ and has both input $C_0$ and a carry output $C_2$. Do not use half adders or full adders, but instead use a two-level circuit plus inverters for the input variables, as needed. Design the circuit by starting with the following equations for each of the two bits of the adder.

\[
S_0 = A_0B_0C_0 + A_0B_0'C_0' + A_0'B_0C_0 + A_0'B_0'C_0 = \text{odd}(A_0, B_0, C_0)
\]

\[
C_1 = A_0B_0 + A_0C_0 + B_0C_0 = \text{majority}(A_0, B_0, C_0)
\]

\[
S_1 = \text{odd}(A_1, B_1, C_1) = \text{odd}(A_1, B_1, \text{majority}(A_0, B_0, C_0))
\]

\[
= A_1B_1 \text{majority}(A_0, B_0, C_0) + A_1'B_1 \text{majority}(A_0, B_0, C_0)'
+ A_1'B_1' \text{majority}(A_0, B_0, C_0) + A_1'B_1' \text{majority}(A_0, B_0, C_0)'
\]

\[
= (A_1B_1 + A_1'B_1') \text{majority}(A_0, B_0, C_0) + (A_1B_1' + A_1'B_1) \text{majority}(A_0, B_0', C_0')
\]

\[
= (A_1B_1 + A_1'B_1') (A_0B_0 + A_0C_0 + B_0C_0) + (A_1B_1' + A_1'B_1) (A_0B_0' + A_0'C_0' + B_0'C_0')
\]

\[
= A_1B_1A_0B_0 + A_1B_1A_0C_0 + A_1B_1B_0C_0 + A_1'B_1A_0B_0 + A_1'B_1B_0A_0C_0 + A_1'B_1'B_0B_0C_0
+ A_1'B_1A_0B_0'C_0 + A_1'B_1A_0'C_0' + A_1'B_1B_0'C_0 + A_1'B_1'B_0'C_0'
\]

\[
C_2 = \text{majority}(A_1, B_1, C_1) = \text{majority}(A_1, B_1, \text{majority}(A_0, B_0, C_0))
\]

\[
= A_1B_1 + A_1 \text{majority}(A_0, B_0, C_0) + B_1 \text{majority}(A_0, B_0, C_0)
\]

\[
= A_1B_1 + A_1(A_0B_0 + A_0C_0 + B_0B_0) + B_1(A_0B_0 + A_0C_0 + B_0B_0)
\]
4. This problem is to design a simple 4 bit arithmetic unit. Your circuit will have two data inputs $A = a_3a_2a_1a_0$ and $B = b_3b_2b_1b_0$ and a data output $X = x_3x_2x_1x_0$. It will also have a control input $C = c_2c_1c_0$ which determines what operation your circuit will perform.

- if $C = 0$ then the output $X$ is equal to input $A$
- if $C = 1$ then the output $X$ is equal to input $B$
- if $C = 2$ then the output $X$ is equal to the 2's complement of input $A$
- if $C = 3$ then the output $X$ is equal to the 2's complement of input $B$
- if $C = 4$ then the output $X$ is equal to $A + B$ where $A$ and $B$ are interpreted as unsigned integer values in the range $0..15$
- if $C = 5$ then the output $X$ is equal to $A + B$ where $A$ and $B$ are interpreted as signed integer values in the range $-8..+7$ (2s complement)
- if $C = 6$ then the output $X$ is equal to $A - B$ where $A$ and $B$ are interpreted as signed integer values in the range $-8..+7$ (2s complement)
- if $C = 7$ then the output $X$ is equal to $B - A$ where $A$ and $B$ are interpreted as signed integer values in the range $-8..+7$ (2s complement)

In addition, your circuit will have a status output $V$ which is 1, if the requested operation results in an invalid result. For example, negating an input whose value is $-8$ should cause the $V$ output to go high. An unsigned addition of values $8$ and $13$ should also cause the $V$ bit to go high.
As a first step in designing your circuit, draw a block diagram that includes a 4 bit adder, one or more 1's complement circuits, incrementers and multiplexors. Include a control block that the C input connects to and that generates the V output and the control signals for the other components. Show each circuit component as a block labeled by its function (e.g. adder) and with all inputs and outputs clearly labeled.

Write logic equations for each output of the control block that will cause the circuit as a whole to implement the specification given above.

Complete your design using the schematic editor. Use only AND, OR, EXOR gates and inverters. On the printout of the schematic, outline in pencil the gates that correspond to the blocks in your block diagram and label them as in your block diagram. Arrange your schematic so that it corresponds as closely as possible to your block diagram.

Simulate your design for the following inputs and verify that the results are correct.

C = 0, A = 5; C = 1, B = 10; C = 2, A = 6; C = 3, B = 8; C = 4, A = 5, B = 14; C = 4, A = 5, B = 7; C = 5, A = 6, B = 5; C = 5, A = 3, B = 4; C = 6, A = 5, B = 2; C = 6, A = 4, B = 6; C = 6, A = 4, B = 6; C = 6, A = 2, B = 3; C = 7, A = 5, B = 6; C = 7, A = 2, B = 6

C = 6, A = 4, B = -6; C = 6, A = 2, B = 3; C = 7, A = 5, B = 6; C = 7, A = 2, B = 6

C = 6, A = 4, B = -6; C = 6, A = 2, B = 3; C = 7, A = 5, B = 6; C = 7, A = 2, B = 6
Table for control signals:

<table>
<thead>
<tr>
<th>C</th>
<th>nA</th>
<th>pA</th>
<th>nB</th>
<th>pB</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>X</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>0</td>
<td>sA·sX</td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>sB·sX</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>cX</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>(sA=sB)·(sA≠sX)</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>(sA≠sB)·(sA≠sX)</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>(sA≠sB)·(sB≠sX)</td>
</tr>
</tbody>
</table>

nA = (C=2) + (C=7)  
pA = C_2 + C_0'

nB = (C=3) + (C=6)  
pB = C_2 + C_0

V = (C=2)·sA·sX + (C=3)·sB·sX + (C=4)·cX + (C=5)·(sA ⊕ sB)’·(sA ⊕ sX) + (C=6)·(sA ⊕ sB)·(sA ⊕ sX) + (C=7)·(sA ⊕ sB)·(sB ⊕ sX)
5. Draw (by hand) a schematic diagram for a 2 input, 4 output decoder using simple 2 input gates. Label each input, output and gate. Write a structural VHDL specification that directly implements the circuit in your schematic. Simulate your decoder on all possible inputs. Write a dataflow VHDL specification of the decoder and simulate this version as well. Hand in the schematic, the VHDL listings and the waveforms from your simulation runs.
library IEEE;
use IEEE.std_logic_1164.all;

entity hw5_5 is
  port ( s1, s0: in STD_LOGIC;
            d0, d1, d2, d3: out STD_LOGIC
  );
end hw5_5;

architecture hw5_5_arch of hw5_5 is

component INV
  port(i: in STD_LOGIC; o: OUT STD_LOGIC);
end component;

component AND2
  port(i0,i1: in STD_LOGIC; o: OUT STD_LOGIC);
end component;

signal s0b, s1b: STD_LOGIC;

begin
  i0: INV port map(i => s0, o => s0b);
  i1: INV port map(i => s1, o => s1b);
  a0: AND2 port map(i0 => s1b, i1 => s0b, o => d0);
  a1: AND2 port map(i0 => s1b, i1 => s0, o => d1);
  a2: AND2 port map(i0 => s1, i1 => s0b, o => d2);
  a3: AND2 port map(i0 => s1, i1 => s0, o => d3);
end hw5_5_arch;
library IEEE;
use IEEE.std_logic_1164.all;

entity hw55b is
    port ( s1, s0: in STD_LOGIC;
           d0, d1, d2, d3: out STD_LOGIC
    );
end hw55b;

architecture hw55b_arch of hw55b is
begin
    d0 <= (not s1) and (not s0);
    d1 <= (not s1) and ( s0);
    d2 <= ( s1) and (not s0);
    d3 <= ( s1) and ( s0);
end hw55b_arch;