Combinational Logic

1. The “majority function” \( m(a, b, c) \) is true when most of the inputs are true, i.e. when any two are true or when all three are true. Fill in the truth table for this function, derive a minimum sum-of-products formula, and show the logic diagram for the resulting reduced equation.

\[
\begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 \\
\end{array}
\]

\[
m(a, b, c) = ab + ac + bc
\]

![Logic Diagram for Majority Function](image)

2. Repeat problem 1, using the “parity function” \( p(a, b, c) \) which is true when an even number of inputs are true, i.e. when zero inputs or two inputs are true.

\[
\begin{array}{cccc}
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 \\
1 & 1 & 1 & 0 \\
\end{array}
\]

\[
p(a, b, c) = \overline{abc} + \overline{abc} + \overline{abc} + \overline{abc}
\]

![Logic Diagram for Parity Function](image)

3. Show how the majority function and parity function can be implemented at the same time using a PLA.

\[
\begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 \\
1 & 1 & 1 & 0 \\
\end{array}
\]

![Logic Diagram for Combined Functions](image)
4. A multiplexor is a combinational circuit that acts like a filter. A 2 x 1 multiplexor has three inputs: two data inputs A and B, and a control input named C. There is one output, named F. When C = 0, the output is whatever value, 0 or 1, is currently on the A input. When C = 1, the output is whatever value is currently on the B input.

- Draw the truth table for this circuit.
- Write a boolean equation that describes F in terms of the inputs A, B, and C. You can use a K-map if you like, but whatever method you use, explain the resulting formula. Does it make sense?
- Draw the logic diagram for the 2 x 1 multiplexor based on your formula.

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Consider two cases:

* if c is 1, the equation reduces to 
  \[ f = b \] 
  or, since \( b \cdot 1 = 1 \), to 
  \[ f = b \]

* if c is 0, the equation reduces to 
  \[ f = a \]

Thus when \( c = 1 \) the output is the current value of \( b \), and when \( c = 0 \) the output is the current value of \( a \).

Sequential Logic

5. If you replace the NOR gates in an RS latch with NAND (Not-AND) gates you will get a circuit that behaves just like the original latch: one output will be the opposite of the other, one combination of inputs is the “stable” combination (it produces no change in the outputs), and two other input combinations can be used to flip the outputs back and forth. Draw the truth table for a NAND gate, the logic diagram for an RS latch using two NAND gates, and explain how it works: what are the stable inputs, what inputs cause \( Q = 0 \), what input causes \( Q = 1 \), and what input combination should not be used?

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>( ab )</th>
<th>R</th>
<th>S</th>
<th>Q</th>
<th>Q*</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Stable is \( R = S = 1 \)
To set \( Q = 1 \) set \( S \) to 0
To set \( Q = 0 \) set \( R \) to 0
If \( R = S = 0 \) then \( Q = Q^* = 1 \)
6. You work for ACME Solar Water Heaters, Inc, and you are given the assignment of designing the digital control logic for the new Model 2000 hot water system. The system has two water heaters: a solar water heater generates hot water when there is enough sunlight, otherwise a backup gas heater is used. Your circuit will control the flow of water from the two heaters into a tank that is connected to the home. To keep the system from over-reacting, the logic is driven by a .1 Hz clock; every 10 seconds you should check to see if the tank level is low, and if it is, open up one of two valves, depending on whether there is enough sun to get water from the solar heater. The input and output signals are:

- An input named $L$ that will be true when the tank level is low.
- An input named $S$ that will be true when there is enough sunlight to run the solar heater.
- An output named $V_S$; when it is true, the valve controlling flow from the solar heater is open.
- An output named $V_G$; when it is true the valve from the gas heater is open.

Design the controller for this system. Draw a finite-state diagram that shows the possible states and the transitions between them, the state transition table, and a logic diagram using edge-triggered D flip-flops.

**Note:** $L = L(S + \overline{S})$

**State Transition Table**

<table>
<thead>
<tr>
<th>$L$</th>
<th>$S$</th>
<th>$Q_0$</th>
<th>$Q_1$</th>
<th>$D_0$</th>
<th>$D_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>00</td>
<td>X</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>00</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>11</td>
<td>X</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>00</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>00</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>00</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Logic Diagram**

Note: Since $Q_0Q_1 = 11$ can never occur, put X’s in this column

### Logic Diagram

- $Q_0Q_1$ can never occur
- $D_0 = LS$
- $D_1 = \overline{L\overline{S}}$

Note: $V_G = Q_0Q_1$, but if $Q_0 = Q_1 = 1$ is impossible then $V_G = Q_1$.
Datapaths

The object of this problem is to show which parts of a single-cycle datapath are used during the execution of SPIM instructions. The diagram on the following page has the outline of the datapath used as an example in the textbook. Your job is to show which components and pathways are involved in the execution of different types of instructions. By “component” we mean things like ALUs, registers, multiplexors, control logic, etc; “pathways” means both data and control lines connecting components.

Print out three copies of the datapath, or drop by the CIS office to pick up three already-printed copies. Indicate the components that are used by darkening the corresponding box or oval in the diagram, and show a path is used by tracing over the path. The program counter and the path leading from the program counter to the instruction memory are already filled in to get you started.

Neatness counts!

- On one copy of the diagram, show which components and paths are used in the execution of a “branch on equal” instruction, assuming the branch is taken.
- On a second copy, darken the components and paths used in the execution of an “add immediate” instruction.
- On a third copy, darken the components and paths used in the execution of a “load word” instruction.
beq

[12 points]
addi

[12 points]
lw

[12 points]