1. Latch/Flip-flop Construction and Properties (10 points): Show how to make an SR latch using only NOR gates

![SR latch diagram]

3. Adders (15 points) Write the equation for the carry into the 4th adder cell in an ALU using carry-lookahead, in terms of P’s and G’s. 
   \[ \text{Cin3} = \text{Cout2} = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0 \text{Cin} \]

4. Combination Circuit design (15 points): A three-variable logic function that is equal to 1 if any two or all three of its variables are equal to 1 is called a majority function. Design a minimum-cost circuit that implements this majority function. Please label your AND gates with the expression they are testing.

![Majority function circuit diagram]

5. Worst Case Path (15 points): Given the following full adder circuit.

![Full adder circuit diagram]

Assuming you have made a 3-bit ripple carry adder using these cells, what is the worst case path through the adder? In other words, how long does it take for the answer to be correct in all cases? Use the following delay values, and assume all input signals become valid at time 0:

- 2-input AND: 6 ns
- 2-input OR: 4 ns
- 2-input XOR: 7 ns

\[
\begin{align*}
\text{Cin1} &= 7 \text{ (xor)} + 6 \text{ (and)} + 4 \text{ (or)} = 17 \text{ns} \\
\text{Cin2} &= 17 + 6 \text{ (and)} + 4 \text{ (or)} = 27 \text{ns} \\
\text{Cin3} &= 27 + 6 \text{ (and)} + 4 \text{ (or)} = 37 \text{ns}.
\end{align*}
\]
6. Sequential Circuit Design (90 points)
   a. State Transition Diagram (20 points)
      I wish to create a double input, $x_i, y_i$, single output Mealy machine that will set the output high whenever $x_i = y_i$ for at least two of the previous three pairs of inputs, e.g. $(1,1) (0,0) (1,1) (1,1) (0,1) (0,0) (1,0) (1,1)$ would produce an output of 0, 1, 1, 1, 1, 0, 1. Draw the state transition diagram of the machine. You may assume that the machine automatically starts in the initial state, $S_i$. Please give your states descriptive subscripts, where $M =$ match, and $D =$ different.

   ![State Transition Diagram](image)

   b. Minimization of States (15 points)
      Use the Partition Minimization Procedure to minimize the states pictured in the state diagram.

   ![Minimized State Diagram](image)

   $P_1 = (i, 1, 2, 3, 4, 5, 6)$
   $P_2 = (i, 4) (1, 2, 3, 5, 6)$
   $P_3 = (i, 4) (2) (1, 3, 5, 6)$
   $P_4 = (i) (4) (2) (1) (3, 5, 6)$

   c. State Code Word Optimization (10 points)
      Based on the above state diagram, and ignoring your minimization in part b, fill in the table to optimize the state code words. State $i$ is already in (0,0)

<table>
<thead>
<tr>
<th>AB</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
</tr>
<tr>
<td>01</td>
<td>i</td>
</tr>
<tr>
<td>11</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>01</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>00</td>
<td>3</td>
</tr>
</tbody>
</table>
d. State Transition Table (25 points)
Given the following state transition diagram, fill in the state transition table. Note that the states are not in numerical order.

<table>
<thead>
<tr>
<th>Present State</th>
<th>Binary Code</th>
<th>Present State</th>
<th>Input</th>
<th>Next State</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>i</em></td>
<td>000</td>
<td>0 0 0</td>
<td>0</td>
<td>0 1 1 0</td>
<td></td>
</tr>
<tr>
<td><em>i</em></td>
<td>000</td>
<td>0 0 0</td>
<td>1</td>
<td>0 0 1 0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>0 0 1</td>
<td>0</td>
<td>1 0 0 1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>0 0 1</td>
<td>1</td>
<td>1 0 0 1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>010</td>
<td>0 1 0</td>
<td>0</td>
<td>1 0 0 1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>010</td>
<td>0 1 0</td>
<td>1</td>
<td>1 1 1 1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>011</td>
<td>0 1 1</td>
<td>0</td>
<td>0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>011</td>
<td>0 1 1</td>
<td>1</td>
<td>0 1 0 0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>1 0 0</td>
<td>0</td>
<td>1 0 0 1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>1 0 0</td>
<td>1</td>
<td>1 1 1 1</td>
<td></td>
</tr>
</tbody>
</table>

Based on the above transition state table, fill in the Karnaugh Maps for A’ and z, and provide the Boolean expression for each.

\[
A' = AB + C
\]

\[
z = \overline{B} \overline{w} + Bw
\]
f. Circuit Implementation (25 points)

Given the following Karnaugh maps fill in the circuit on the next page that uses a DFF for A, and a SRFF for B.

\[
\begin{array}{c|cccc}
A' & AB & 00 & 01 & 11 & 10 \\
\hline
w & 0 & 1 & 1 & d & 0 \\
& 1 & 1 & 1 & d & 0 \\
\end{array}
\]

\[
\begin{array}{c|cccc}
\bar{A}' & AB & 00 & 01 & 11 & 10 \\
\hline
w & 0 & 0 & 0 & d & 1 \\
& 1 & 1 & 1 & d & 1 \\
\end{array}
\]

\[
\begin{array}{c|cccc}
z & AB & 00 & 01 & 11 & 10 \\
\hline
w & 0 & 0 & 0 & d & 1 \\
& 1 & 0 & 0 & d & 0 \\
\end{array}
\]

\[
\begin{align*}
A' &= \bar{A} \\
S &= A + w \\
R &= \bar{A} \bar{w} \\
z &= \bar{A} w + \bar{A} B w
\end{align*}
\]