CS 151
Quiz 2

Name : __________________ , __________________
      (Last Name)                (First Name)

Student ID : _______________

Signature : _______________

Instructions:

1. Please verify that your paper contains 6 pages including this cover.
2. Write down your Student-Id on the top of each page of this quiz.
3. This exam is closed book. No notes or other materials are permitted.
4. Total credits of this midterm are 40 points.
5. To receive credit you must show your work clearly.
6. No re-grades will be entertained if you use a pencil.
7. Calculators are NOT allowed.
Q1: [Mux/Decoder application] [15 points]

For function $F(x,y,z) = xy'z + xyz' + x'y'z'$:

(a) Create the truth table (2 points)

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
<th>$Z$</th>
<th>$F$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</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>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</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>0</td>
</tr>
</tbody>
</table>

(b) Implement $F$ by means of a 4-to-1 multiplexer (5 points)

(HINT: You can use NOT gates to inverse the input to the MUX)

![4-to-1 Multiplexer Diagram]
(c) A 3-to-8 decoder (4 points)

\[
F = S_1'S_0'I_0 + S_1'S_0I_1 + S_1S_0'I_2 + S_1S_0I_3
\]

To implement this function we totally need 16 AND gates and for the output we need two
additional OR gates. So we totally need 18, 2-input gates.

(d) Compare the two design in parts (b) and (c) based on the number of equivalent 2-input gates (4 points)

(You don’t have to count the number of NOT gates used in the input of MUX)

For 4-to-1 MUX the output equation is:

\[
F = S_1'S_0'I_0 + S_1'S_0I_1 + S_1S_0'I_2 + S_1S_0I_3
\]

To implement this function we totally need 8 AND and 3 OR gates which becomes 11 2-input gates.

For 3-to-8 Decoder the output equation is:

\[
D_0 = I_2'I_1'I_0', \quad D_1 = I_2'I_1'I_0, \quad D_2 = I_2'I_1I_0', \quad D_3 = I_2'I_1I_0, \quad D_4 = I_2'I_1'I_0', \quad D_5 = I_2'I_1I_0', \quad D_6 = I_2'I_1I_0, \quad D_7 = I_2'I_1I_0
\]

To implement the decoder we need 16 AND gates and for the output we need two additional OR gates. So we totally need 18, 2-input gates.
Q2: [Combinational design] [10 points]

We want to design a combinational circuit that computes the function \( f(x) = x^2 + x + 2 \) for a 2-bit \( x \):

a. How many bits do we need for output? (2 points)

\( x \) varies between 0 (00) and 3 (11) so \( f(x) \) varies between: 2 and 14. To be able to represent these values in binary we need 4 bits.

b. Draw the truth table for this function. (4 points)

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( x_0 )</th>
<th>( y_3 )</th>
<th>( y_2 )</th>
<th>( y_1 )</th>
<th>( y_0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

c. Using the truth table in part(b), write the equations for representing the output function (no need for simplification). (4 points)

\[
\begin{align*}
y_3 &= x_1 x_0' + x_1 x_0 \\
y_2 &= x_1' x_0 + x_1 x_0 \\
y_1 &= x_1' x_0' + x_1 x_0 \\
y_0 &= 0
\end{align*}
\]
Q3: [Latch analysis] [10 points]

Shown below is a NAND implementation of gated D-latch:

\[ D \quad C \quad Q \quad Q' \]

The timing diagrams of D and C are shown below. Show the timing diagram for Q and Q': (Assume that Q=0 at t0 and there is no gate delay)

At the very first of the execution, Q and Q’ are actually both 0 and because of that the circuit is instable and both Q and Q’ fluctuate between 0 and 1, however when C goes down, D forces Q to become 1 and Q’ to become 0. Now the circuit is stable. The question should’ve been designed in a way that the circuit was stable from the very first point.

<table>
<thead>
<tr>
<th>C</th>
<th>D</th>
<th>Q</th>
<th>Q'</th>
</tr>
</thead>
<tbody>
<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>1</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>Q_{prev}</td>
<td>Q'_{prev}</td>
</tr>
</tbody>
</table>
Q4: [FSM design] [5 points]

Draw the FSM diagram for a counter that counts like this:

7 → 6 → 5 → 4 → 3 → 2 → 1 → 0 → 7 → 6 ……

The counter should have an Enable input. When the Enable = 1 it counts as shown above, if Enable = 0, it stops counting.