General Statement: In this exam, we assume that every input comes with its complement.

I. Adders/Subtracters: A Half Subtractor inputs two bits \( x \) and subtrahend \( y \), and outputs the difference \( d \) and borrow-out \( b_{out} \), while a Full Subtractor inputs three bits \( x \), subtrahend \( y \) and borrow-in \( b_{in} \), and outputs the difference \( d \) and borrow-out \( b_{out} \).

I.1. Write the truth tables of the Half Subtractor and the Full Subtractor. Describe those logic using minimal sum of products expressions.

I.2. Implement a Full Subtractor with only one Half Subtractor and a minimal two-level NOR gates. Draw the logic diagram.-
II. Serial Adders:
II.1. A sequential adder inputs $a_i, b_i$, the $i$'th bit of two binary numbers in each clock cycle for $i = 0$ to $n - 1$ and outputs the sum $s_i$. Implement the adder with a T flip-flop, and a minimal AND .OR-NOT network (if the network is needed). Write the state table to describe the next states and outputs. Draw the schematic diagram.

Class Notes

II.2. A sequential three-at-a-time adder inputs $a_i, b_i, e_i$, the $i$'th bit of three binary numbers in each clock cycle for $i = 0$ to $n - 1$ and produces the sum $s_i$. Implement the adder with a minimal number of Full Adders and two D flip-flops. Other logic components are not needed, nor available. Draw the logic diagram.
III. Counters: A Modulo-8 Counter reads data (D2, D1, D0), load LN, count enable CNT, and generates state (Q2,Q1,Q0) and terminal count TC. Design a counter that counts from 1815 to 2009 using four Modulo-8 Counters and minimal AND-OR-NOT networks. Draw the schematic diagram.
IV. Design a counter with a repeated output sequence 0, 1, 6, 7, 2, 3, 4 with a Modulo-8 Counter and a minimal AND-OR-NOT network. Write the Boolean expression and draw the schematic diagram.
V. System Designs:
Implement the following algorithm:
Alg(X, Y, Z, start, U, done);
Input X[7:0], Y[7:0], start;
Output U[7:0], done;
Local-object A[7:0], B[7:0], C[7:0];
S1: If start' goto S1;
S2: A<= X || B<= Y || C<= (00001111) || done<= 0;
S3: C<=Inc(C) || If C'[7] then A<= Add(A, B) else A<= Add(A, -B);
S4: B<= Inc(B) || If B'[7] goto S3;
S5: U<= A || done<= 1 || goto S1;
End Alg

V.1. Design a data subsystem that is adequate to execute the algorithm.

V.1.1. Use a table to list the instructions and corresponding components for data path subsystem.

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Comp</th>
<th>Reg</th>
</tr>
</thead>
<tbody>
<tr>
<td>A&lt;= X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B&lt;= Y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C&lt;= (00001111)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C&lt;= Inc(C)</td>
<td></td>
<td>Inc</td>
</tr>
<tr>
<td>A&lt;= A+B</td>
<td>Add</td>
<td></td>
</tr>
<tr>
<td>A&lt;= A-B</td>
<td>Add</td>
<td></td>
</tr>
<tr>
<td>B&lt;= Inc(B)</td>
<td>Inc</td>
<td></td>
</tr>
<tr>
<td>U&lt;= A</td>
<td>Wires</td>
<td></td>
</tr>
</tbody>
</table>
V.1.2. Draw the schematic diagram to show the data path subsystem. Label the inputs, outputs, and control signals of all components.

V.2. Design the control subsystem.
V.2.1. Use a table to list the value of control signals for every state.
V.2.2. Draw the state diagram.

V.2.3. Implements the control subsystem with a one hot encoding design. Draw the logic diagram.