Please read the following instructions carefully: The exam contains 6 problems of which we are free to choose four or more to answer. The grade will be counted according to the best four. This is an open book final. Web searches are encouraged. If there is any uncertainty about the problems, make and state your assumptions. The final should be done individually. Therefore, no discussion or collaboration is allowed. Please submit your solution as a pdf file via TED.

1. Two Level Logic Minimization: Given a switching function with on-set $F = \{f_i\}$, don’t-care set $D = \{d_i\}$ and off-set $R\{r_i\}$, where each $f_i$, $d_i$, or $r_i$ is a product term. In addition we have all the prime implicants $P = \{p_i\}$. We want to identify the essential prime implicants for a two level logic minimization.

1.1. State the definition of the essential prime implicant.

1.2. Suppose that you are the chief software engineer. We need a high level pseudo code to identify that prime implicant $p_i$ is essential. Write the pseudo code and explain your algorithm. Hint: Assume that you have some low level routines available for the pseudo code. Some (not all) possible routines are listed below. You may define and use similar routines in your algorithm.

adjacency($p_i$, $p_j$): returns true if product terms $p_i$ and $p_j$ are adjacent, otherwise returns false.

consensus($p_i$, $p_j$): returns the consensus if the two product terms $p_i$ and $p_j$ are adjacent, otherwise returns an empty set.

tautology($S$): The set $S$ contains a list of product terms. This routine returns true if the sum of all product terms in the set $S$ is always true for any input combination, otherwise it returns false.

2. Shannon’s Expansion: This problem is about the proof and application of Shannon’s expansion.

2.1. Prove the Shannon’s expansion theorem of switching algebra.

2.2. Given a nine variable switching function:

$$F(a, b, c, d, e, f, g, h, i) = abd + afg + ahi + abf + a'bg + a'bi + a'fh + a'fi + cdf + cd'g + cd'h + cdi + c'd'f + c'dg + c'dh + c'd'i' + ace + b'eg + dcf + d'eh + ae'i + e'gh + e'h'i + e'gi'$$

Find one binary assignment of the input variables so that function $F = 0$. Explain your derivation if you can or cannot find a solution.
3. The logic diagram below shows a 4-bit random sequence generator. Answer the following questions to describe the functional behavior of the random sequence generator.

![4-bit random sequence generator](image)

**Figure 1: 4-bit random sequence generator**

3.1. Suppose that the flip-flops are reset initially, list all states \((Q_3, Q_2, Q_1, Q_0)\) in sequence.

3.2. The timing characteristics of the components are summarized below.

- Flip-flop: clock-to-Q maximum delay \(t_{peq} = 40\) ps, clock-to-Q minimum delay \(t_{ccq} = 30\) ps, setup time \(t_{setup} = 50\) ps, hold time \(t_{hold} = 20\) ps
- Logic gate (each AND, OR, Inverter, XNOR): propagation delay \(t_{pd} = 35\) ps, contamination delay \(t_{cd} = 15\) ps.

3.2.1. Derive the minimum clock period.

3.2.2. Draw the timing diagram of \(Q_3\) for the first 6 clock cycles.

4. You have been enlisted to design a soda machine dispenser for your department lounge. Sodas cost only 25 cents. The machine accepts nickels, dimes and quarters. When enough coins have been inserted, it dispenses the soda and returns any necessary change. Design an FSM controller for the soda machine. The FSM inputs are Nickel, Dime and Quarter, indicating which coin was inserted. Assume that exactly one coin is inserted on each cycle. The outputs are Dispense, ReturnNickel, ReturnDime and ReturnTwoDimes. When the FSM reaches 25 cents, it asserts Dispense and the necessary Return outputs required to deliver the appropriate change. Then it should be ready to start accepting coins for another soda.

5. Gray codes have a useful property in that consecutive numbers differ in only a single bit position. The Gray code representation for the numbers 0 to 7 is given in Table 1. Design a 3-bit modulo 8 Gray code counter FSM with no inputs and three outputs. ( A modulo \(N\) counter counts from 0 to \(N - 1\), then repeats. For example a watch uses a modulo 60 counter for the minutes and seconds that counts from 0 to 59). When reset the output should be 000. On each clock edge, the output should advance to the next Gray code. After reaching 100, it should repeat with 000.
<table>
<thead>
<tr>
<th>Number</th>
<th>Gray Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>1</td>
<td>0 0 1</td>
</tr>
<tr>
<td>2</td>
<td>0 1 1</td>
</tr>
<tr>
<td>3</td>
<td>0 1 0</td>
</tr>
<tr>
<td>4</td>
<td>1 1 0</td>
</tr>
<tr>
<td>5</td>
<td>1 1 1</td>
</tr>
<tr>
<td>6</td>
<td>1 0 1</td>
</tr>
<tr>
<td>7</td>
<td>1 0 0</td>
</tr>
</tbody>
</table>

Table 1: 3-bit Gray code

6. System Design: Follow the IEEE 754 single format with a 32-bit word size. Let us assume that the two numbers are positive and ignore the subnormal cases and round off errors. Design a floating point addition using a datapath and control subsystem methodology.
6.1 Describe your algorithm.
6.2 Describe the data subsystem with a schematic diagram.
6.3 Draw the state diagram of the control subsystem and list the control signals generated by the control subsystem.