1. (4 points) Given Boolean function

\[ F(a, b, c, d) = a'b'c + ab'd' + abc'd' + b'c'd \]

with don’t care conditions \( bc'd \)

Find all the prime implicants (PIs) and essential prime implicants (EPIs). You can use any method.

2. (3 points) Find the Boolean expression that describe the following circuit and then minimize this expression.

![Circuit Diagram]

3. (5 points) Use full adder, half adder, and basic logic gates (AND, OR, NOT only) to compute the expression: \( 1ABC - A'1C' \)

For example:

\[
\begin{align*}
1101 - 010 &= 1011 \\
1100 - 011 &= 1001 \\
1011 - 110 &= 0101 \\
\end{align*}
\]

Assuming that the cost of a basic logic gate is 1, a half adder is 2, and a full adder is 4, find the design with the minimum cost. Draw the logic diagram of your design and specify the cost of your design.

4. (4 points) Given two 8-bit numbers \( A_7A_6A_5A_4A_3A_2A_1A_0 \) and \( B_7B_6B_5B_4B_3B_2B_1B_0 \), use two shift registers (and any other logic gates or combinational components if necessary) to construct and store two new 8-bit numbers \( L_7L_6L_5L_4L_3L_2L_1L_0 \) and \( S_7S_6S_5S_4S_3S_2S_1S_0 \), where \( L_1 = \max(A_1, B_1) \) and \( S_1 = \min(A_1, B_1) \).

For example, when inputs \( A=1010,0000 \) and \( B=1100,0011 \), the new numbers will be \( L=1110,0011 \) and \( S=1000,0000 \) and will be stored in two registers.

5. (4 points) Draw the state diagram of a system that recognizes the occurrence of sequence 10110. For example, on input sequence \( 0101,1011,0011 \) the corresponding output should be \( 0000,0100,1000 \)

You only need to draw the state diagram, no need to implement the circuit.
1. 6 PIs: \( ac' \), \( c'd \), \( a'b'd \), \( ab'd' \), \( b'cd' \), \( a'b'c \)
   1 EPI: \( ac' \)

2. \( F = xy + (x'y' + z)' + z + wz = xy + (x + y)z' + z = xy + x + y + z = x + y + z \)

3. \( 1ABC - A'1C' = 1ABC + 1A0C + 1 = AXY1 \)
   where \( XY = B+C \) can be obtained by one HA with cost 2. \( X \) is the carry, \( Y \) is the sum.

4. Store the two 8-bit numbers \( A_7A_6A_5A_4A_3A_2A_1A_0 \) and \( B_7B_6B_5B_4B_3B_2B_1B_0 \) in two shift registers \( RA \) and \( RB \).
   Shift out one bit from each registers at each clock cycle, \( A_i \) and \( B_i \), define \( L_i \) and \( S_i \) as follows:
   \[ L_i = \max(A_i, B_i) = A_i + B_i \]
   which can be implemented with a logic OR
   \[ S_i = \min(A_i, B_i) = A_i \cdot B_i \]
   which can be implemented with a logic AND
   Feed \( L_i \) and \( S_i \) into the two shift registers \( RA \) and \( RB \).
   After 8 clock cycles, \( RA \) and \( RB \) will store the two new 8-bit numbers \( L_7L_6L_5L_4L_3L_2L_1L_0 \) and \( S_7S_6S_5S_4S_3S_2S_1S_0 \).

5. \( S0 \ (0) \rightarrow \ S0 \quad S0 \ (1) \rightarrow \ S1 \)
   \( S1 \ (0) \rightarrow \ S2 \quad S1 \ (1) \rightarrow \ S1 \)
   \( S2 \ (0) \rightarrow \ S0 \quad S2 \ (1) \rightarrow \ S3 \)
   \( S3 \ (0) \rightarrow \ S2 \quad S3 \ (1) \rightarrow \ S4 \)
   \( S4 \ (0) \rightarrow \ S2 \ (1) \quad S4 \ (1) \rightarrow \ S1 \)

Format: \( \text{Current\_state (input)} \rightarrow \text{next\_state (output, 0 if not specified)} \)