1) Let: \( F(v,w,x,y,z) = \sum m(9,11,12,13,14,15,19,20,28,30) \)

Execute the Quine-McCluskey algorithm to find all the Prime Implicants of the function

The algorithm has been started for you below (minterms are listed in increasing 1's count order). Complete the algorithm and circle the prime implicants.

<table>
<thead>
<tr>
<th>( m )</th>
<th>Index Order</th>
<th>✓</th>
<th>1-Cubes</th>
<th>✓</th>
<th>2-Cubes</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>01001</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>01100</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>10100</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>01011</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>01101</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>01110</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>10011</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>11100</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>01111</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>11110</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2) A set of seven Prime Implicants were generated for a Boolean function. The cover table below is the n derived for this function. Use this cover table and categorize the PIs into (i) Essential PI(s) (EPI), (ii) Less-than PI(s) (LTPI), (iii) Secondary Essential PI(s) (SEPI), or (iv) Redundant PI(s) (RPI).

<table>
<thead>
<tr>
<th>Prime Imp</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>A4</th>
<th>A5</th>
<th>A6</th>
<th>A7</th>
<th>PI category</th>
</tr>
</thead>
<tbody>
<tr>
<td>PI1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PI2</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PI3</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>PI4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>PI5</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PI6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>PI7</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Did the function above have any don’t cares? Explain your reasoning.
3) A combinational circuit is to be designed according to the following specification:
The inputs are $A_3A_2A_1A_0$, and the output is $Z$. The four-bit inputs represent a 4-bit binary number $A$. When $4 \leq A \leq 5$, or $11 \leq A \leq 14$, $Z = 0$. When, $1 \leq A \leq 2$, or $8 \leq A \leq 10$, $Z = 1$. Otherwise, the output is of no concern.

Find the corresponding K-map of $Z(A_3, A_2, A_1, A_0)$

Represent $Z(A_3, A_2, A_1, A_0)$ in **sum of product** standard form with **minimum number of literals**.
4) Decomposition has been performed transforming $F_1$ into $F_2$. Carry out the decomposition and determine the values of $G_1$ and $G_2$ in terms of $A$, $B$, $C$, and $D$.

$$F_1 = AC + B'D + AD + B'C$$
$$F_2 = G_1\cdot G_2$$

5) The logic diagram of a combinational logic circuit is given below:

Express the corresponding Boolean function in the **product of Maxterm** format:
6) Convert the following logic schematic diagram into two separate realizations; one using only NOR gates, and one using only NAND gates. You may use two-input NOR/NAND gates and inverters. Assume the complements of all input Boolean variables are NOT available.

NOR-only Realization:

NAND-only Realization:
7) Implement the following functions by placing X’s at the appropriate PLA wire junctions.

\[ F_1 = AB + A'C + BC \]
\[ F_2 = (A'C + AB'C')' \]
8) For each of the circuits below, indicate whether the output of the circuit together with its input variables will have even or odd parity.

\[ P_1 \]

\[ P_2 \]

\[ P_3 \]

\[ P_4 \]
9) Design a 4-16 line decoder using a 3-to-8 line decoder, a 1-to-2 line decoder, and 16 2-input AND gates
10) A combinational circuit is defined by the following three Boolean functions:

\[ F_1 = a \bar{e} + \bar{b}c \]
\[ F_2 = (\bar{b} + \bar{c})(\bar{a} + c) \]
\[ F_3 = \prod M(0, 2, 4, 6, 7) \]

Design the circuit with a 3-to-8 decoder and three NOR gates. No inverter or any other type of logic gates are allowed. Do NOT assume the complement of any Boolean variables are available either.
11) Use the 8-to-1 multiplexer below to implement an exclusive or function for four bits. This is also known as the "odd" function. The function \( \text{exor}(w, x, y, z) \) is to be:

\[
\text{exor}(w, x, y, z) = w \oplus x \oplus y \oplus z
\]

Draw the circuit by factoring out the variable “w” in the space below. You may use only NOT, OR, and AND gates.
13) You are to implement an adder that adds a constant 1 to a four-bit number $A = A_3A_2A_1A_0$ and uses as few Half-Adders (HA) as possible. You are provided sufficiently many of HAs in the figure below and make the necessary connections in the figure. Note that after adding a 1 to $A$ we can get a 5-bit sum.

One of these HAs can be simplified further. Identify the part that can be simplified and write its simplified equations (for both sum and carry signals).
14) In a 4-bit carry look-ahead adder, the carry out bit $C_4$ can be expressed as:

$$C_4 = G_{0-3} + P_{0-3}C_0 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0$$

where $G_{0-3} = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0$ is called a group generate function and $P_{0-3} = P_3P_2P_1P_0$ is called a group propagate function. Suppose we use four of such carry look-ahead adders to perform addition of 16-bit binary numbers. We have $C_4 = G_{0-3} + P_{0-3}C_0$, $C_8 = G_{4-7} + P_{4-7}C_4$, and $C_{12} = G_{8-11} + P_{8-11}C_8$.

(a) (9 points) Derive a two-level SOP realization for $C_8$ and $C_{12}$ in terms of $C_0$ and group generate and group propagate functions. We will ignore the carry out bit $C_{16}$.

Assume that an XOR gate contributes 2 gate delays. It takes 2 gate delays to evaluate $P_3s$ and $G_3s$ in each 4-bit carry look-ahead adder. What is the maximum gate delay to compute the result using the 16-bit hierarchical carry look-ahead adder described above?
15) Two four-bit unsigned (integer) numbers \( A = A_3A_2A_1A_0 = 0101 \), and \( B = B_3B_2B_1B_0 = 0110 \) are to be added.

For the \( i \)-th bit, its carry-in is denoted by \( C_i \), carry-out is denoted by \( C_{i+1} \), generate function denoted by \( G_i \) and propagate function denoted by \( P_i \).

Suppose that the XOR gate is implemented such that it has a propagation delay of \( 1.5 \mu s \), and all other AND, OR, or INVERTER gates all have a gate propagation delay of \( 1 \mu s \). (\( 1 \mu s = 10^{-6} \) seconds). Assume \( \{A_i; B_i; i = 0; 1; 2; 3\} \) and \( C_0 \) are available simultaneously at time \( t = 0 \).

(a) (6 points) The values of \( C_3 \) = \[ \square \]. \( P_2 \) = \[ \square \]; \( G_1 \) = \[ \square \].

If the addition is performed by a *ripple carry adder* where each adder is realized with a partial full adder (PFA) and the carry is evaluated using the formula \( C_{i+1} = G_i + P_i \cdot C_i \).

(b) (3 points) \( C_2 \) will be available is at \( t = \) \[ \square \] \( \mu \) seconds.

Now suppose that a four-bit carry-look-ahead adder is to be used to carry out this addition. Answer the following questions:

(c) (3 points) At the earliest, \( P_3 \) will be available at \( t = \) \[ \square \] \( \mu \) seconds.

(d) (3 points) At the earliest, \( S_3 \) will be available at \( t = \) \[ \square \] \( \mu \) seconds.
16) The Verilog for a structural circuit definition of module "what_is_this" is shown below. Convert the structural definition to a logic diagram of the circuit. Draw the logic diagram.

```verilog
module what_is_this(A, B, w, x, y, z, F);
    input A, B, w, x, y, z;
    output F;
    wire A_n, B_n, p1, p2, p3, p4;
    not
        g0(A_n, A);
        g1(B_n, B);
    nand
        g2(p1, A_n, B_n, w);
        g3(p2, A_n, B_n, x);
        g4(p3, A, B_n, y);
        g5(p4, A, B, z);
        g6(F, p1, p2, p3, p4);
endmodule
```