1) FPGA Placement and Routing

2) MUX-based design

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</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>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Maps to a binary tree:

Each node is a decision variable

Output entries of truth table
function computed here is \( e \cdot a + c \cdot 1 = c \)

this line computes \( \overline{c} \)

3 MUX
2 INV.

A valid solution
This solution can be further improved.

\[ \overline{bc} + bc \]
\[ \text{XOR}(b,c) \]

\[ \overline{bc} + bc \]
\[ \text{XNOR}(b,c) \]

\[ \text{2MUX} \]
\[ \text{2INV, better solution} \]
\[ F = ab + ac + bc. \]

\[
F = \overline{ab + ac + bc}
\]

\[
= \overline{ab} \overline{ac} \overline{bc}
\]

\[
= (\overline{a+b}) (\overline{a+c}) (\overline{b+c})
\]

\[
= (\overline{a+b+c}) (\overline{b+c})
\]

\[
= \overline{a} \overline{b} + \overline{a} \overline{c} + \overline{b} \overline{c}
\]

\[
= F(\overline{a}, \overline{b}, \overline{c})
\]

For majority function, inverting the inputs leads to the same string as inverting the output. However, this isn't always the case.

Let \( F(a, b) = a \cdot b \).

\[ F(\overline{a}, \overline{b}) = \overline{ab} = \overline{a} + \overline{b} \]

\[ F(\overline{a}, \overline{b}) = \overline{a} \cdot \overline{b} \] not the same.
3) i)  

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>01</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>00</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ F = C + B'D' + A'BD \]

ii)  

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>01</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>00</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ F = A'C + B'CD + A'B'D \]
iii) $F = BD + AC + B'C$

iv) 

$F = CE + AB'E + BC'D'E' + A'C'DE'$
\( F = A'B'D'E' + ABD'E + AB'D \)

4) i)

\[ F = A'D + C'D \]

\( F = B'C' + B'D' + BCD \)
5) Using K-maps

Prime Implicants: $C'D'$, $BC'$, $A'CD$, $AB'D$, $A'BD$, $AC'$, $B'CD$

Essential Prime Implicant = $C'D'$

The simplified function is given by $C'D' + BC' + A'CD + AB'D$

There are 4 product terms and the literal cost is 10.
\[ f = \overline{ace} + \overline{bce} + abce + \overline{a}bce + \overline{abcde} + \overline{abcd} + abc \overline{d} \]

26 literals, in minimized SOP

In decomposed form:

\[ g_0 = \overline{abc} + \overline{abc} + abc \quad (9 \text{ literals}) \]
\[ g_1 = \overline{abc} + abc \quad (6 \text{ literals}) \]
\[ h = \overline{g_0} \overline{g_1} e + g_0 g_1 d + g_0 g_1 e \quad (9 \text{ literals}) \]
No simplification

Already simplified

Now consider don't cares @ input of h

Primary inputs \{g, b, c, d, e\}.
\[ g_0, g_1, \] depend on \(g, b, c\).
For all values of \(g, b, c\),
\[ g_0 \oplus g_1 = 1, 1 \] never possible
So, \( g_0 = 1 \land g_1 = 1 \) = don't care.

\[ \text{don't care cube} = g_0 g_1 \]

Now simplify \( h \) w/ \{g_0 g_1 = 11\}

\[ \text{as don't care} \]

\[ h = \overline{g_0 g_1} e + g_0 \overline{d} + g_1 e \]

Boolean decomp \( \rightarrow \) creates don't cares

Simplify logic further.

\[ (7 \text{ literals}) \]

\[ + 9 + 6 \]

\[ g_0 \overset{7 \text{ literals}}{\rightarrow} g_1 \]

\[ = 22 \text{ literals} \]