1) \[ f(a, b, c, d) = b' \cdot d' + c' \cdot d \cdot a + a' \cdot b \cdot d \]

2) \[ f(a, b, c, d) = b' \cdot d' + c' \cdot d \cdot b + a \cdot c' \cdot d \]

3) \[ f(a, b, c, d) = b' \cdot d' + c' \cdot d \cdot b + c' \cdot a \cdot b \]
A set of gates is said to be universal if any combinational system can be implemented using gates from that set. So for example \( \{\text{NAND}\} \) is a universal set. Any of \( \{\text{AND, OR, NOT, NAND, NOR, XOR, XNOR}\} \) can be implemented using only NAND gates.

b) We need to see if the following gates can be implemented in each of the cases: \( \{\text{AND, OR, NOT, NAND, NOR, XOR, XNOR}\} \)

i) \( \{\text{NAND, NOR}\} \)

\[
\text{AND} = \begin{array}{c}
\text{\(a\)} \\
\text{\(b\)}
\end{array} \\
\text{NOR} = \begin{array}{c}
\text{\(a\)} \\
\text{\(b\)}
\end{array}
\]

\[
\text{NOT} = \begin{array}{c}
\text{\(a\)}
\end{array}
\]

\[
\text{OR} = \begin{array}{c}
\text{\(a\)} \\
\text{\(b\)}
\end{array}
\]

AND from above
b) \[ \text{XOR} = a'\cdot b + a\cdot b' \]

This can be implemented using the AND, NOT & or gates seen earlier.

Thus this set \( \{\text{NAND, NOR}\} \) is universal.

ii) \( \{\text{XOR, NOT}\} \)

\[
\begin{array}{c|cc}
 a & b & a\oplus b \\
\hline
 0 & 0 & 0 \\
 0 & 1 & 1 \\
 1 & 0 & 1 \\
 1 & 1 & 0 \\
\end{array}
\]

We see that any modifications to XOR only yields a NOT gate. Inverting XOR leaves us with another combination of AND's & OR's -

\[
(a\oplus b)' = (a'\cdot b + a\cdot b')' = (a+b')(a'+b)
\]

Hence this set is NOT universal.

iii) The single gate we have here is \( f(x,y) \)

i.e. let us denote this module as

\[
\begin{array}{c}
y \downarrow M \rightarrow x'y \\
\hline
\text{NOT:} \hspace{1cm} \begin{array}{c}
x \downarrow M \rightarrow x'
\end{array}
\end{array}
\]
\text{AND:} \quad x \quad \xrightarrow{\text{NOT}} \quad \begin{array}{c} M \\ \rightarrow \end{array} \quad \rightarrow x \cdot y

\text{NAND:} \quad \text{Since we have AND & NOT we have NAND}

\text{OR:} \quad x \quad \xrightarrow{\text{NOT}} \quad \rightarrow \begin{array}{c} M \\ \rightarrow \end{array} \quad \rightarrow \quad \rightarrow x + y

\text{Since NOT & AND have been seen, we can do OR}

\text{At this point we could conclude that all gates can be derived since we have AND, NAND, OR & NOT, which form a universal set in more than 1 way.}

\text{IV:} \quad x \quad \xrightarrow{\begin{array}{c} M \\ \rightarrow \end{array}} \quad \rightarrow \quad (x+y) \quad \rightarrow

\text{NOT:} \quad \begin{array}{c} 1 \\ \rightarrow \end{array} \quad y \quad \xrightarrow{\begin{array}{c} M \\ \rightarrow \end{array}} \quad \rightarrow \quad \rightarrow \quad \rightarrow z

\text{OR:} \quad x \quad \xrightarrow{\begin{array}{c} M \\ \rightarrow \end{array}} \quad \rightarrow \quad (x+y)

\text{NOR:} \quad \text{Since we have OR & NOT, we have NOR.}

\text{Since } \forall \text{NOR? is a universal set, } \begin{array}{c} M \\ \rightarrow \end{array} \text{ is}
\[ f(a, b, c) = \sum m(0, 1, 2) + \sum d(6, 7) \]

<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>1</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>0</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>

*Getting the 1's terms and 'don't cares':*

- 0 0 0 0
- 1 0 0 1 \(\Rightarrow (0, 1) 00\)
- 2 0 1 0 \(\Rightarrow (0, 2) 00\)
- 6 1 1 0 \(\Rightarrow (2, 6) 00\)
- 7 1 1 1

Notice we'll skip (6, 7) since they are both 'don't cares'.

\[ \begin{array}{cccccc}
0 & 1 & 2 & 6 & 7 \\
(0, 1) & X & X & & & \\
(0, 2) & X & X & X & & \\
(2, 6) & X & X & X & & \\
\end{array} \]
So we keep $(0,1)$ and $(0,2)$ since these two terms cover all the 1's.

\[ f = a'c' + a'b' \]

\[
\begin{align*}
Q_0(t+1) &= Q_0 \oplus (x'Q_1) & \text{(1)} \\
Q_1(t+1) &= Q_1 \oplus (x + Q_0) & \text{(2)}
\end{align*}
\]

\[ y(t) = Q_0 + Q'_0 \]

We get the above two relationships because we know the network is using T-flip. The characteristic equation of T-flip is

\[ Q(t+1) = T(t) \oplus Q(t) \]

b) Using first two equations, we create the state table.

<table>
<thead>
<tr>
<th>( \bar{Q}_0 )</th>
<th>( \bar{Q}_0 )</th>
<th>0</th>
<th>1</th>
<th>( y(t) )</th>
<th>( S )</th>
<th>( x )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>10</td>
<td>10</td>
<td>1</td>
<td></td>
<td></td>
<td>A</td>
</tr>
<tr>
<td>0 1</td>
<td>00</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td>B</td>
</tr>
<tr>
<td>1 0</td>
<td>01</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td>C</td>
</tr>
<tr>
<td>1 1</td>
<td>01</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td>D</td>
</tr>
</tbody>
</table>

Let \( A = 00 \), \( B = 01 \), \( C = 10 \), \( D = 11 \)
* Note in part (a) $Q_0Q_1$ are switched to their correct order $Q_3Q_0$. So the solution may look a little different from the discussion section on Thursday. This solution is correct!

5. Implement JK using T flip-flop.

$$Q(t+1) = T(t) + Q(t) = Q(t)\bar{F} + \overline{Q(t)}J$$

Xor'ing both sides with $Q(t)$ we get

$$T(t) = (Q\bar{F} + \overline{QJ}) \oplus Q$$

Since we want a minimal expression using AND, NOT, OR gates, we can minimize the expression using K-maps.
From above k-map we get

\[ T = KQ + J\overline{Q} \]