Problem 1

a) | Q_4 | Q_3 | Q_2 | Q_1 | Q_0 |
---|-----|-----|-----|-----|-----|
S_0 | 0   | 0   | 0   | 0   | 0   |
S_1 | 1   | 0   | 0   | 0   | 0   |
S_2 | 1   | 1   | 0   | 0   | 0   |
S_3 | 1   | 1   | 1   | 0   | 0   |
S_4 | 1   | 1   | 1   | 1   | 0   |
S_5 | 1   | 1   | 1   | 1   | 1   |
S_6 | 0   | 1   | 1   | 1   | 1   |
S_7 | 0   | 0   | 1   | 1   | 1   |
S_8 | 0   | 0   | 0   | 1   | 1   |
S_9 | 0   | 0   | 0   | 0   | 1   |

\[ T_{CD0} \]
\[ T_{CD0} = 1 \text{ in } S_9 \text{ and up and } CE = 1 \]
\[ \text{or in } S_0 \text{ and down and } CE = 1 \]
There are 10 states, then if coded in one-hot it is required 10 D-FF and if coded in binary it is required 4 D-FF.

<table>
<thead>
<tr>
<th>CE</th>
<th>UD-L</th>
<th>Current-State</th>
<th>Next-State</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>S0</td>
<td>S0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>S0</td>
<td>S1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>S0</td>
<td>S0</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
<td>S1</td>
<td>S1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>S1</td>
<td>S2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>S1</td>
<td>S1</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
<td>S2</td>
<td>S2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>S2</td>
<td>S3</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>S2</td>
<td>S1</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
<td>S3</td>
<td>S3</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>S3</td>
<td>S0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>S3</td>
<td>S2</td>
</tr>
</tbody>
</table>
d) Main VHDL sentence of CC2:

```vhdl
CC2 : Process ( Current-state, CE, UD-L )
BEGIN
  IF ( ( Current-state = S0 ) AND ( CE = '1' ) AND ( UD-L = '1' ) ) THEN
    TC01 <= '1';
  ELSE
    TC01 <= '0';
  END IF
  CASE Current-state IS
    WHEN S0 =>
      Q <= "00000";
    WHEN S1 =>
      Q <= "10000";
    WHEN S2 =>
      Q <= "11000";
    WHEN S3 =>
      Q <= "00001";
    WHEN S4 =>
      Q <= "00010";
    WHEN S5 =>
      Q <= "00100";
    WHEN S6 =>
      Q <= "01000";
    WHEN S7 =>
      Q <= "10000";
    WHEN S8 =>
      Q <= "00001";
  END CASE
END PROCESS
```
e) The functional simulation is ideal
   In this case the maximum frequency can be \( \infty \)

\[
\tau_{\text{max}} = \frac{1}{t_{\text{CO}} + N \cdot t_{\text{PD}}}
\]

If \( C=2 \) has three levels of gates

\[
\tau_{\text{max}} = \frac{1}{4.8 \text{ns} + 3 \cdot 1.7 \text{ns}} = 103.09 \text{ MHz}
\]
Problem 2

This circuit can be used to implement a Johnson counter of 4 bits.

Two VHDL files will be required.
Problem 3

(a) 
\[ T = \left( \frac{25 \text{ Hz}}{1} \right) \rightarrow 40 \text{ ms} \]

\[ T_{\text{off1}} = T - T_{\text{on1}} \]

\[ T_{\text{on1}} = 20\% \]

\[ T = 40 \text{ ms} \]

\[ DC_1 = 20\% \]

\[ T_{\text{on2}} = 32 \text{ ms} \]

\[ T_{\text{off2}} = 8 \text{ ms} \]

\[ T_{\text{on1}} = 8 \text{ ms} \]

\[ T_{\text{off1}} = 32 \text{ ms} \]

(b) 
Accordingly to the port pins selected,

\[
\begin{array}{cccccccc}
\text{TRISA} & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
\text{PORTA} & x & x & x & x & x & x & x \\
\text{TRISD} & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
\text{PORTD} & a & b & c & d & e & f & g \\
\end{array}
\]

\[
\begin{array}{cccccccc}
\text{TRISB} & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\text{PORTB} & x & x & x & x & x & x & x \\
\end{array}
\]

\[ 0 \rightarrow \text{output is the default value} \]

\[
\begin{array}{cccccccc}
\text{INT0IE} = 1 \\
\text{TMODIE} = 1 \\
\end{array}
\]
The TMRO module is as represented approximately in this picture:

\[
\frac{f_{osc}}{4} \Rightarrow \text{select fosc/4}
\]

\[
\text{selected prescaler}
\]

\[
\text{different numbers } N_1 = 2, 4, 8, \ldots, 256
\]

\[
0.5 \mu s \leftarrow \text{Time base}
\]

\[
\text{TMRO1F when the system overflows}
\]

\[
\text{Timing period} = \left(\frac{4}{f_{osc}}\right) \cdot N_1 \cdot N_2
\]

\[
T_{on_1} = 8 \text{ ms} = 8000 \mu s = (0.5 \mu s) \cdot N_1 \cdot N_2
\]

\[
(T_{on_2})
\]

\[
T_{off_1} = 32 \text{ ms} = 32000 \mu s = (0.5 \mu s) \cdot N_1 \cdot N_2
\]

\[
(T_{off_2})
\]

In this way, we can load the TMRO with the same value

\[
N_2 = 256 \Rightarrow \text{TMRO} = (256 - 250) = 6
\]

and depending on the prescaler, the 2 periods are possible. Swapping \( N_1 \rightarrow N_1 = 64 \rightarrow \text{TMRO1F every 8 ms} \)

\[
N_1 = 256 \rightarrow \text{TMRO1F every 32 ms}
\]

In this way, the mechanism is:

\[
\text{TMRO1F = 1, TMRO1F = 1, TMRO1F = 1, TMRO1F = 1, TMRO1F = 1, TMRO1F = 1, TMRO1F = 1, TMRO1F = 1}
\]

\[
\text{f = 25 Hz, DC = 20\%}
\]
The hardware detects overflow and sets the flags which are transformed into convenient variables to run the FSM.

The ASCII value, like A, B, C, D, E.

This value will set the N1 (64).

This value will set the N1 (256).

When reloading the TMRO.
Read input

- Read PORTA
- Mask 0b 0000 0000
- Save in Var-DC

Var-DC ≠ 0

<table>
<thead>
<tr>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Var-TON = ?</td>
<td>Var-TON = 32</td>
</tr>
<tr>
<td>Var-TOP = 32</td>
<td>Var-TOP = 8</td>
</tr>
</tbody>
</table>

Write output 1

- Shift left 7 bits

| Bi1 = Var.W < 5 |
| Write PORTA = Bi1 |
| Write PORTD = Var.Seg |
| End |

This number range will program the prescaler of the TDO

```
TDO 0
INTOF = 1

<table>
<thead>
<tr>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>B_Flag = 1</td>
<td></td>
</tr>
</tbody>
</table>

| TLQOF = 1 |
| No |
| End |

TM22 = (256 \cdot N2)

\[ N2 = 250 \]

Start the on-period

<table>
<thead>
<tr>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Var.W = 0</td>
<td></td>
</tr>
</tbody>
</table>

Pre scaler = Var-TON (n1)

<table>
<thead>
<tr>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre scaler = Var-TOP (n1)</td>
</tr>
<tr>
<td>TMR0.Flag = 1</td>
</tr>
</tbody>
</table>

Start the off-period

<table>
<thead>
<tr>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre scaler = Var-TOP (n1)</td>
</tr>
<tr>
<td>TMR0.Flag = 1</td>
</tr>
</tbody>
</table>
Output logic truth table

<table>
<thead>
<tr>
<th>Var_PC</th>
<th>Current_state</th>
<th>Var_Beg</th>
<th>Var_W</th>
<th>Var_Ton</th>
<th>Var_Toff</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Idle</td>
<td>0x01</td>
<td>0</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>0</td>
<td>Start Timer</td>
<td>0x00</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Start Timer</td>
<td>0x00</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>Wave on</td>
<td>0x30</td>
<td>1</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>1</td>
<td>Wave on</td>
<td>0x6D</td>
<td>1</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>0</td>
<td>Wave OFF</td>
<td>0x30</td>
<td>0</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>1</td>
<td>Wave OFF</td>
<td>0x6D</td>
<td>0</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>x</td>
<td>Stop Timer</td>
<td>0x00</td>
<td>0</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>
state_logic_2() truth table

<table>
<thead>
<tr>
<th>B_Flag</th>
<th>TMRO Flag</th>
<th>Current State</th>
<th>Current State +</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>x</td>
<td>Idle</td>
<td>Idle</td>
</tr>
<tr>
<td>1</td>
<td>x</td>
<td>Idle</td>
<td>Start Timer</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>Start Timer</td>
<td>Wave_ON</td>
</tr>
<tr>
<td>x</td>
<td>0</td>
<td>Wave_OFF</td>
<td>Wave_ON</td>
</tr>
<tr>
<td>x</td>
<td>1</td>
<td>Wave_OFF</td>
<td>Wave_OFF</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Wave_OFF</td>
<td>Wave_ON</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Wave_OFF</td>
<td>Stop Timer</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>Stop Timer</td>
<td>Idle</td>
</tr>
</tbody>
</table>

This table is for setting all the state transitions.

This is the way the main program is organised → FSM