Problem 1

1. The truth table, the symbol and an example of timing diagram

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
<th>Timing Diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

The circuit looks like a simple 2-bit multiplier with no extra signals for expansion.

<table>
<thead>
<tr>
<th>A(1, 0)</th>
<th>B(1, 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

\[ A(1, 0) \times B(1, 0) = R(3, 0) \]

\[ \text{Min. Pulse} = 1.5 \mu s \]

When using the VHDL simulation tool, if we assume all the vectors with a Min. Pulse duration: 16 × Min. Pulse = 24 μs to test the complete table and be able to verify it.

2. Canonical forms to represent logic functions

\[ R_2 = \bar{g}(B, A) = \sum_{i} \bar{m}(10, 11, 14) \]

\[ R_1 = f(B, A) = \prod M(0, 1, 2, 3, 4, 5, 7, 10, 12, 15) \]

For instance \[ m_{10} = B, B', A, A' \]

\[ M_x = (B_1 + B_3 + A_1 + A_0) \]

\[ \phi \leq \psi \]
3. Method of decoders

Internally, it can be built using other components of the same kind, like Dec-3.3, or it can be a flat circuit. Let's suppose it is a flat circuit.

The decoder is a circuit which generates all the mixtures for the input D(3..0), so that we simply have to OR some of them for implementing functions.

\[
\begin{align*}
R_3 &= m_{15} \\
R_2 &= \Sigma m(10, 11, 14) \\
R_1 &= \Sigma m(6, 7, 9, 11, 13, 14) \\
R_0 &= \Sigma m(5, 7, 13, 15)
\end{align*}
\]

Thus, if the component Dec-3.16 is a flat circuit, the design will consist of 2 VHDL source files.
4. Method of multiplexers using a MUX 4

Here, the good idea is to copy again the truth table so that we can inspect it and section it in order to see which functions have to be connected to the channels.

<table>
<thead>
<tr>
<th>Channel</th>
<th>Ch0</th>
<th>Ch1</th>
<th>Ch2</th>
<th>Ch3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The partial functions are:

- \( f_0 = f(A_1, A_0) = '0' \) all ways to 0
- \( f_1 = f(A_1, A_0) = A_1 \) a buffer of the input port
- \( f_2 = f(A_1, A_0) = A_0 \)
- \( f_3 = f(A_1, A_0) = A_1 \oplus A_0 \)
  \( \text{or } (A_1'A_0 + A_1A_0') \)
  \( \text{or } (A_1 + A_0)(A_1' + A_0') \)

In the same way, we can implement the other outputs. A MUX is required to solve an output. So, to implement the complete Multi-2bit 4 MUX-4 and some logic gates are required.
5. Flat circuit using logic gates and minimized equations

\[ \begin{array}{c|c}
\text{BBAA} & \text{RRRR} \\
1010 & 3210 \\
\end{array} \]

\[ \begin{array}{c|c}
1111 & 1...
1-10 & .1..
101- & .1..
-110 & ..1.
1-01 & ..1.
10-1 & ..1.
011- & ..1.
-1-1 & ...1 \\
\end{array} \]

If the minimized output table format comes from a SOP:

\[ R_3 = B \cdot B_0 \cdot A_1 \cdot A_0 \]

\[ R_2 = B \cdot A \cdot A_0' + B_1 \cdot B_0' \cdot A \]

\[ R_1 = B_0 \cdot A \cdot A_0' + B_1 \cdot A_0' \cdot A + B_0' \cdot A_0 + B_1 \cdot B_0' \cdot A \]

\[ R_0 = B \cdot A_0 \]

in VHDL:

\[ R_0 \equiv B_1 \text{ and } A_0 \]

\[ R_3 \equiv B_0 \text{ and } A_1 \text{ and } \neg(A_0) \text{ or } B_1 \text{ and } \neg(A_1) \text{ and } A_0 \text{ or } B_1 \text{ and } \neg(B_0) \text{ and } A_0 \text{ or } \neg(B_1) \text{ and } B_0 \text{ and } A_1 \]
6. Using only NOR to solve digital circuits

From the Minilog output table:

\[ R_3 = B \cdot B_0 \cdot A_1 \cdot A_0 = (B \cdot B_0 \cdot A_1 \cdot A_0)'' \]

\[ R_3 = (B'_1 + B'_0 + A'_1 + A'_0)' \]

\[ \uparrow \]

NOR of 4 inputs that can be implemented using NOR of 2 inputs in this way:

\[ \left( \left( B'_1 + B'_0 \right)' \cdot \left( A'_1 + A'_0 \right)' \right)' = R_3 \]

NOR gate with inputs \( B'_1, B'_0, A'_1, A'_0 \) and output \( R_3 \) (labeled as \( R_3 = (x + y)' \))

\[ \uparrow \]

The NOT can be implemented also using NOR-2

\[ A_0 \rightarrow \overrightarrow{A_0'} \]
7. Calculating the maximum speed of computing and the circuit's power consumption

\[ \bar{I_{CC}} = 5V \]
\[ I_{CC} = \frac{I_{CCH} + I_{CCL}}{2} = \frac{11mA + 4.2mA}{2} \]
\[ I_{CC} = 2.65mA \]

Power consumption = 18 gates \times \left( 5V \cdot 2.65mA \right) = \frac{P_c}{\text{gate}} \]
\[ P_c = 18 \times 13.25\text{mW} = 238.5\text{mW} \]

Maximum speed of processing is \( \frac{1}{L_p \times m_s} = \frac{1}{3 \text{ level} \times (11ns + 8ns)} = 17.54 \text{MHz} \)

This is the worst case output waveform. For example, \( R_2 \), when switching \( 4 \) and \( 6 \) at maximum speed.
Problem 2

1. The design of a 7-bit adder/subtractor was studied in PG.

2. Using 2C and 7 bit the range is \(-2^6 \leq Q \leq 2^6 - 1\)
   \[-64 \leq Q \leq 63\]

   The overflow flag can be deduced as the XOR logic function between the last two carries \((C_6 \oplus C_7 = \text{OV})\) of the 7-bit internal arithmetic adder.

   ![Diagram of 7-bit adder/subtractor](image)

   \(\text{OV} \rightarrow \text{demonstrated using several examples in PG class notes}.

3. Let's try several operations:

   a) \(A = (+39)_{10} \quad B = (00101010)_2\)
   \[OP = \oplus \quad 0 \quad 54\]
   \(\Rightarrow \text{The result is -54 and there is no overflow}\)
   \(Z = \emptyset\)

   ![Addition of binary numbers](image)

   \(C_7 = \emptyset\)

   \(-15 \Rightarrow \text{OK!}\)

   b) \((0010\ 0110)_2 - (-55)_{10}\)
   \[A = +22\]
   \[B = -55\]
   \[R = A - B\]
   \[OP = 1\]
   \(\Rightarrow \text{The result is -77 \text{ out of range} \Rightarrow \text{OV} = 1}\)

   The computer will solve the subtraction in this way
   \[R = A - B = A + 2C(B)\]

   ![Subtraction with carry](image)

   \(C_7 = \emptyset\)

   \[-51 \Rightarrow \text{OV} = 1\]
c) \((+18)_{10}\)  \(\text{oP} = 0\)  \(B = (11011110)_{2C}\)  \(-18\)

\[\begin{array}{c}
0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 1 \\
0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0
\end{array}\]

The result: \(R = A + B = 0\)

\[\begin{array}{c}
0 \text{v} = 0 \\
Z = 1
\end{array}\]

Let's see: \(\begin{array}{cccc}
& 0 & 0 & 1 & 0
\end{array}\)
\(\begin{array}{cccc}
1 & 1 & 0 & 1 & 1 & 1
\end{array}\)

\[\begin{array}{c}
0 \ 0 \ 0 \ 0 \ 0 \ 0
\end{array}\]  \(\rightarrow\)  \(R = 0\)  \(\checkmark\)

d) \((-31)_{10}\)  \(- (0101110)_{2C}\)  \(+22\)

\[\begin{array}{c}
1 \ 1 \ 0 \ 1 \ 0 \ 1
\end{array}\]

The result: \(-31 - (+22) = -53\)

\[\begin{array}{c}
0 \text{v} = 0 \\
Z = 0
\end{array}\]

\[\begin{array}{cccc}
& 0 & 0 & 1 & 1 & 1
\end{array}\]
\(\begin{array}{cccc}
1 & 1 & 0 & 1 & 0 & 1
\end{array}\)

\[\begin{array}{c}
0 \ 0 \ 0 \ 1
\end{array}\]
\(\begin{array}{cccc}
0 & 1 & 1 & 0 & 1 & 0 & 1
\end{array}\]

\(-53\)  \(\text{ok!}\)
4. Timing diagram and VHDL test bench

There are 15 inputs, so \(\text{Min-Pulse} = 10.5 \, \mu s\)

\[ A + B + \text{OP} \]

\[ \Rightarrow \text{run for } 2^{15} \cdot 10.5 \, \mu s = 345 \, ms \]

To translate the stimulus vector to VHDL:

\[
\begin{align*}
A & \Leftarrow "0100111"; \\
B & \Leftarrow "1001010"; \\
\text{OP} & \Leftarrow '1'; \\
\text{wait for Min-Pulse;}
\end{align*}
\]

etc...

So, it is necessary a test time of 345 ms in case of likely to check and verify all the input vectors.

\[ \text{This "mechanical" process can be carried out automatically using VHDL test bench features like } \text{ASSERT} \]
5. Maximum speed of operations

\[ t_p = 2 \text{ns} \]

\[ x0 \lor (\text{adder-16bit}) \cdot 7 + \frac{1}{2} \times 0 \lor \]

\[ \text{3 levels of gates} \]

\[ t_p + 7 \cdot (3 \times t_p) + t_p = 23t_p = 46\text{ns} \]

\[ f_{\text{max}} \leq \frac{1}{46\text{ns}} = 21.74 \text{ MHz} \]

\[ \approx 21.7 \text{ millions of operations per second} \]

6. The EQ flag detector can be implemented using \(\text{N} \times \text{OR} \)

\[ \text{internal architecture based on gates} \]

\[ y = \bar{a}_i \cdot \bar{b}_i + a_i \cdot b_i \]

\[ y = (a_i \oplus b_i)' \]

\[ y = (a_i + b_i) \cdot (a_i + b_i)' \]