Homework 2

(Due date: October 3rd @ 9:30 am)
Presentation and clarity are very important!

Problem 1 (10 pts)
- Complete the following table. Use the fewest number of bits for every case. Show your procedure.

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Sign-and-magnitude</th>
<th>1's complement</th>
<th>2's complement</th>
</tr>
</thead>
<tbody>
<tr>
<td>-257</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>128</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-120</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-61</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-1022</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>511</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Problem 2 (15 pts)
- We need to perform the following operations, where numbers are represented in 2's complement:
  - 73 + 45
  - -35 + 64
  - -129 + 128
- For each case:
  - Determine the minimum number of bits required to represent both summands. You might need to sign-extend one of the summands, since for proper summation, both summands must have the same number of bits.
  - Perform the binary addition in 2’s complement arithmetic. The result must have the same number of bits as the summands.
  - Determine whether there is overflow by:
    - Using $c_{n}, c_{n-1}$ (carries).
    - Performing the operation in the decimal system and checking whether the result is within the allowed range for $n$ bits, where $n$ is the minimum number of bits for the summands.
  - If we want to avoid overflow, what is the minimum number of bits required to represent the summands and the result?

Problem 3 (15 pts)
- Sign-extension in 2’s complement: Whenever we need to increase the number of bits for representing a number, we append the MSB to the left as many times as needed:
  $$b_{n-1} b_{n-2} \ldots b_0 = b_{n-1} \ldots b_{n-1} b_{n-2} \ldots b_0$$
- Examples: 0010112 = 0000 10112 = $2^2 + 2^0 = 5$
  101012 = 1110 1012 = $-2^4 + 2^2 + 2^0 = -2^6 + 2^5 + 2^4 + 2^2 + 2^0 = -11$

We can think of the sign-extended number as an $m$-bit number, where $m > n$:
$$b_{n-1} \ldots b_{n-1} b_{n-2} \ldots b_0 = k_{m-1} \ldots k_n k_{n-1} k_{n-2} \ldots k_0$$

- Demonstrate that $b_{n-1} b_{n-2} \ldots b_0$ represents the same decimal number as $b_{n-1} \ldots b_{n-1} b_{n-2} \ldots b_0$, i.e., demonstrate that sign-extension is correct for any $m > n$.
- Useful formula: $\sum_{i=k}^{r} r^i = \frac{r^{k+1} - 1}{1-r}$, $r \neq 1$

Problem 4 (15 pts)
- Implement the following functions using i) decoders and ii) multiplexers:
  - $F = X + Z + ZY$
  - $F = XY + YZ + XZ$
  - $F(X, Y, Z) = \prod_{i=1}^{7} (M_i, M_4, M_6)$
  - $F = X \oplus Y \oplus Z$
- Using only 2-to-1 MUXs, implement the XOR and XNOR gates.
- Using only a 4-to-1 MUX, implement the following functions.
  - $F(X, Y, Z) = \sum_{i=1}^{7} (m_i, m_3, m_5, m_7)$
  - $F(X, Y, Z) = \sum_{i=1}^{3} (m_i, m_3, m_5)$
  - $F(X, Y, Z) = \sum_{i=5}^{7} (m_5, m_7)$
PROBLEM 5 (10 PTS)

- A 20-bit address line in a µprocessor handles up to \(2^{20} = 1\, MB\) of addresses, each address containing one-byte of information. We want to connect four 256KB memory chips to the µprocessor.
- Sketch the circuit that: i) addresses the memory chips, and ii) enables only one memory chip (via CE: chip enable) when the address falls in the corresponding range. Example: if \(address = 0x5FFF\), \(\rightarrow\) only memory chip 2 is enabled (CE=1). If \(address = 0xD0123\), \(\rightarrow\) only memory chip 4 is enabled.

PROBLEM 6 (15 PTS)

- Complete the timing diagram of the circuit shown below:

- The following VHDL code corresponds to the shaded circuit. Complete the timing diagram:

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity tst is
    port (b, c, d: in std_logic;
          z: out std_logic_vector(1 downto 0));
end tst;

architecture bhv of circ is
    signal x, y: std_logic;
    begin
        process (b, c, d)
        begin
            z <= c & d;
            if b = '1' then
                case d is
                    when '1' => z <= "01";
                    when others => z <= "00";
                end case;
            else
                if c = '0' then z <= "11";
                end if;
            end if;
        end process;
    end bhv;
```

---

Instructor: Daniel Llamocca
Problem 7 (20 pts)

- A straightforward implementation of the multiplication operation (for positive or unsigned numbers) is called **Array Multiplier**: the partial products are added up at every column. The figure depicts the hardware implementation for multiplying two unsigned numbers of 4 bits.

- One of the drawbacks of this approach is the long delay between the input and the output. In the figure, the bits $p_7, p_6, p_5, p_4$ require the inputs to pass through 4 adders as well as AND gates.
- An alternative implementation, called **Wallace Multiplier**, features a shorter delay between the input and the output. The hardware implementation is more complicated though.
- The figure shows a Wallace multiplier implementation for two unsigned numbers of 4 bits. Note how the addition operation of the 4 rows has been rearranged so that only 2 rows are added up at every stage.

✓ Write the VHDL implementation for the multiplication operation of two unsigned numbers of 4 bits using: i) Array Multiplier, and ii) Wallace multiplier. Use the **Structural Description**: create a separate VHDL file for the Full Adder circuit.
✓ Simulate both circuits and make sure that they are performing the correct operation.
✓ Attach your VHDL code, testbench code, and simulation results.