1. (10) State the five axioms and their duals that define the Boolean algebra ($2^1$).

1a. $x = 0$ if $x \neq 1$
1b. $x = 1$ if $x \neq 0$

2a. If $x = 0$, then $\overline{x} = 1$
2b. If $x = 1$, then $\overline{x} = 0$

3a. $0 \cdot 0 = 0$
3b. $1 + 1 = 1$

4a. $1 \cdot 1 = 1$
4b. $0 + 0 = 0$

5a. $0 \cdot 1 = 1 \cdot 0 = 0$
5b. $1 + 0 = 0 + 1 = 1$

2. (10) Given the following truth table for the Boolean function $f$ and its variables a, b, and c, showing all steps in your solution, implement the logic function $f$ as a function of the variables a, b, and c with a logic diagram using NOT, AND, and OR gates.

<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>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</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>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

\[ f = \overline{a} \cdot b \cdot c + a \cdot \overline{b} \cdot \overline{c} \]
3. Given the template VHDL code, complete the code for the architecture declaration using the dataflow style indicated for the logic equation \( f = (\neg x \cdot y \cdot z) + (x \cdot y \cdot \neg z) \) where \( \neg b \Rightarrow \text{not } b \). Don't forget to relate the function variables to the VHDL variables.

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity prob3 is port(
a: in std_logic_vector (2 downto 0);
f: out std_logic);
end prob3;

architecture dataflow of prob3 is begin

\[
\begin{align*}
&f \Leftarrow (\neg a(2) \land a(1) \land a(0)) \lor (a(2) \land a(1)) \land (\neg a(0))
\end{align*}
\]

end dataflow;

b.(10) conditional signal assignment (when-else statement)

architecture dataflow of prob3 is begin

\[
\begin{align*}
f \Leftarrow \begin{cases} '1' & \text{when } (a = "011") \text{ or } (a = "110") \\
'0' & \text{else}
\end{cases}
\end{align*}
\]

end dataflow;

c.(10) selected signal assignment (with-select-when statement)

architecture dataflow of prob3 is begin

\[
\begin{align*}
f \Leftarrow \begin{cases} '1' & \text{when } "011" \\
'1' & \text{when } "110" \\
'0' & \text{when other}
\end{cases}
\end{align*}
\]

end dataflow;

4. (25) Given the template VHDL code, implement the synchronous finite state machine defined by the state diagram given below using the two process PS/NS method. Define any signals that are needed to complete your implementation.

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity prob4 is port (
    x, clk: in std_logic;
    f: out std_logic
);
end prob4;

architecture behavior of prob4 is
    -- should be behavioral; important for tools
    type state_type is (zero, one, two, error);
    signal ps, ns: state_type;

begin
    sync_proc: process (clk)
    begin
        if rising_edge (clk) then
            ps <= ns;
        end if;
    end process sync_proc;

    comb_proc: process (ps,x)
    begin
        f <= '0';
        case ps is
            when zero => f <= '1';
               if x = 'i' then ns <= one;
               else ns <= two;
               end if;
            when one => f <= '0';
               if x = '1' then ns <= two;
               else ns <= zero;
               end if;
            when two => f <= '0';
               if x = '1' then ns <= zero;
               else ns <= one;
               end if;
            when error => f <= '0';
            when others => null;
        end case;
    end process comb_proc;
end behavior;
```
5. The VHDL code for a clocked synchronous finite state machine is given below. Showing all steps and all definitions required in your solution, determine the:
a.(15) draw a logic diagram which implements the state machine with D flip-flops.

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity prob5 is port (  
  x, clk: in std_logic;
  z: out std_logic
);
end prob5;

architecture dataflow of prob5 is
signal d: std_logic;
signal q: std_logic;
begi
  d <= not (q and x);
  z <= (not q) or x;
  q <= d when rising_edge(clk) else q;
end dataflow;
```

b.(10) state diagram indicating whether it is a Mealy or Moore architecture

```
\[ \begin{array}{c|cc}
q & 0 & 1 \\
0 & 1 & 1 \\
1 & 1 & 0
\end{array} \quad \begin{array}{c|cc}
\delta & 0 & 1 \\
0 & s_0 & s_1 \\
1 & s_0 & s_0
\end{array} \quad \begin{array}{c|cc}
\pi & 0 & 1 \\
0 & s_0 & s_0 \\
1 & s_1 & 1
\end{array} \\
\text{(transition/output)} \quad \text{(state/output)} \quad \text{(state assignment)}
```

\[ d = \neg q \cdot x + q \]  
\[ z = \neg q + x \]  
\text{Mealy}