Problem 1:

```
----------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity statemachine is
    Port (clk, rst, x1, x2: in std_logic;
    z1, z2: out std_logic);
end statemachine;

architecture Behavioral of statemachine is
type STATE_TYPE is (S1, S2, S3);
signal CS, NS : STATE_TYPE;
signal x, z : std_logic_vector(1 downto 0);
begin
x <= x1&x2;
process (clk, rst)
begin
    if rst = '1' then
        CS <= S1;
    elsif falling_edge(clk) then
        CS<= NS;
    end if;
end process;

process(clk, rst)
begin
    if rst = '1' then
        NS <= S1;
    elsif falling_edge(clk) then
        case CS is
            when S1 =>
                if x = "00" then
                    NS <= S1;
                else
                    NS <= S3;
                end if;
            when S2 =>
                if x = "00" then
                    NS <= S3;
                elsif x = "01" then
```

```
                    NS <= S2;
                else
                    NS <= S1;
                end if;
            when s3 =>
                if x = "00" then
                    NS <= S2;
                elsif x = "01" then
                    NS <= S1;
                else
                    NS <= S3;
                end if;
            when others =>
                NS <= S1;
        end case;
    end if;
end process;
z <= "01" when CS = S1 else
     "11" when CS = S2 else
     "10";
z1 <= z(0);
z2 <= z(1);

end Behavioral;
```

Problem 2:

```
-------------------------------------------------------------------------------------
----------
T-flip-flop with reset
```

Problem 3:

```
-------------------------------------------------------------------------------------
----------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity bit_counter is
    Port (x : in std_logic_vector(3 downto 0);
          B : out std_logic_vector(2 downto 0));
end bit_counter;
architecture Behavioral of bit_counter is
```

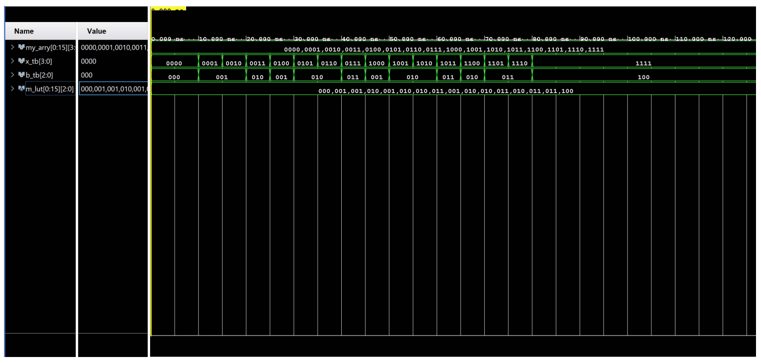
```
signal cnt : unsigned(2 downto 0) := "000";
begin
process(x)
begin
for i in 0 to 3 loop
    if x(i) = '1' then
        cnt <= cnt + 1;
    else
        cnt <= cnt;
    end if;
end loop;
b <= std_logic_vector(cnt);

end process;
end Behavioral;
```

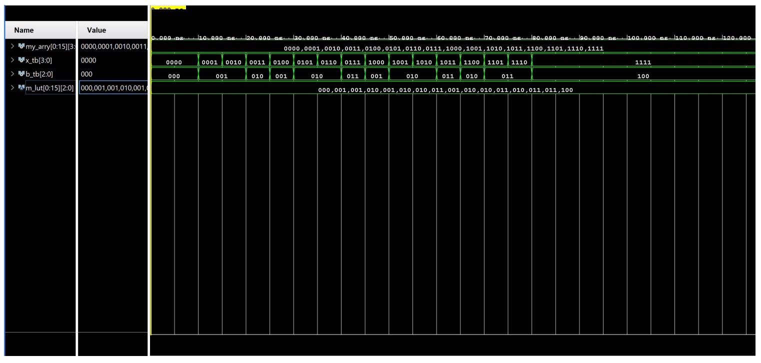
Problem 4:

```
----------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity bit_counter_lt is
    Port (x : in std_logic_vector(3 downto 0);
          B : out std_logic_vector(2 downto 0));
end bit_counter_lt;

architecture Behavioral of bit_counter_lt is
TYPE LUT is array (0 to 15) of std_logic_vector(2 downto 0);
constant my_lut : lut :=(
0 => "000",
1 => "001",
2 => "001",
3 => "010",
4 => "001",
5 => "010",
6 => "010",
7 => "011",
8 => "001",
9 => "010",
10 => "010",
11 => "011",
12 => "010",
13 => "011",
14 => "011",
15 => "100");
begin

b <= my_lut(TO_INTEGER(unsigned(x)));

end Behavioral;
```

Problem 5:

```
----------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity bit_counter_TB is
--  Port ( );
end bit_counter_TB;

architecture Behavioral of bit_counter_TB is
component bit_counter_lt is
    Port (x : in std_logic_vector(3 downto 0);
          B : out std_logic_vector(2 downto 0));
end component;

component bit_counter is
    Port (x : in std_logic_vector(3 downto 0);
          B : out std_logic_vector(2 downto 0));
end component;
TYPE LUT is array (0 to 15) of std_logic_vector(2 downto 0);
constant m_lut : lut :=(
0 => "000",
1 => "001",
2 => "001",
3 => "010",
4 => "001",
5 => "010",
6 => "010",
7 => "011",
8 => "001",
9 => "010",
10 => "010",
11 => "011",
12 => "010",
13 => "011",
14 => "011",
15 => "100");
Type in_vect is array(0 to 15) of std_logic_vector(3 downto 0);
signal my_arry : in_vect :=(
0 => "0000",
1 => "0001",
2 => "0010",
3 => "0011",
4 => "0100",
5 => "0101",
6 => "0110",
7 => "0111",
8 => "1000",
9 => "1001",
10 => "1010",
11 => "1011",
12 => "1100",
13 => "1101",
14 => "1110",
15 => "1111");
signal x_tb : std_logic_vector(3 downto 0) := "0000";
signal b_tb : std_logic_vector(2 downto 0):= "000";

begin
UUT2: bit_counter_lt port map (x_tb, b_tb);
UUT1: bit_counter port map (x_tb, b_tb);
process
begin

for i in 0 to 15 loop
    wait for 5 ns;
    x_tb <= my_arry(i);
    assert b_tb = m_lut(i)
        report "Incorrect response bit_counter";
end loop;
```


Name
Value
my_arry[0:15][3:
0000,0001,0010,0011,
x_tb[3:0]
0000
b_tb[2:0]
000
m_lut[0:15][2:0]
000,001,001,010,001,
0.000 ns
10.000 ns
20.000 ns
30.000 ns
40.000 ns
50.000 ns
60.000 ns
70.000 ns
80.000 ns
90.000 ns
100.000 ns
110.000 ns
120.000
0000,0001,0010,0011,0100,0101,0110,0111,1000,1001,1010,1011,1100,1101,1110,1111
0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
000 001 010 001 010 011 001 010 011 010 011 100
000,001,001,010,001,010,010,011,001,010,010,011,010,011,011,100


Name
Value
my_arry[0:15][3:
0000,0001,0010,0011,
x_tb[3:0]
0000
b_tb[2:0]
000
m_lut[0:15][2:0]
000,001,001,010,001,
0.000 ns
10.000 ns
20.000 ns
30.000 ns
40.000 ns
50.000 ns
60.000 ns
70.000 ns
80.000 ns
90.000 ns
100.000 ns
110.000 ns
120.000
0000,0001,0010,0011,0100,0101,0110,0111,1000,1001,1010,1011,1100,1101,1110,1111
0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
000 001 010 001 010 011 001 010 011 010 011 100
000,001,001,010,001,010,010,011,001,010,010,011,010,011,011,100