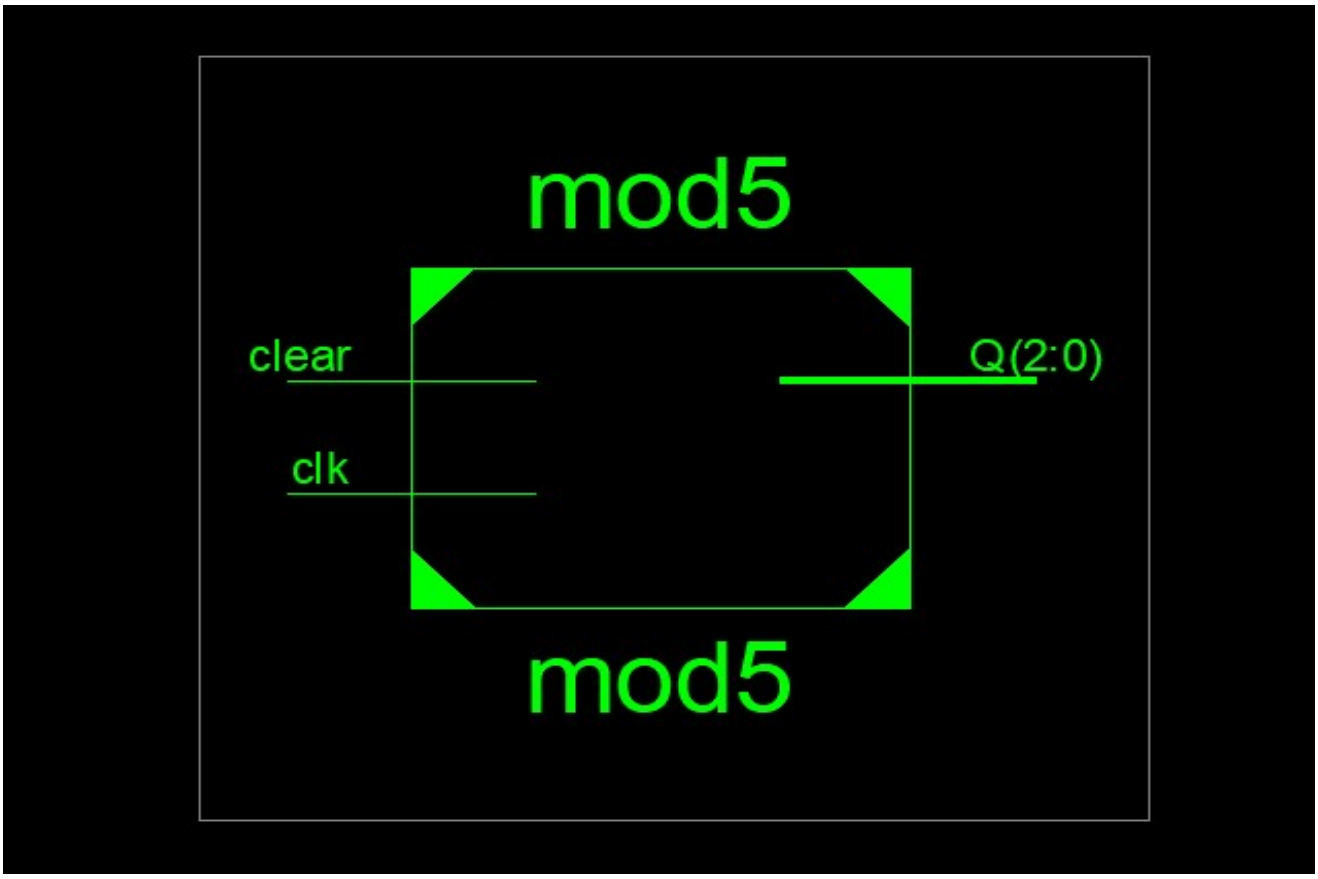
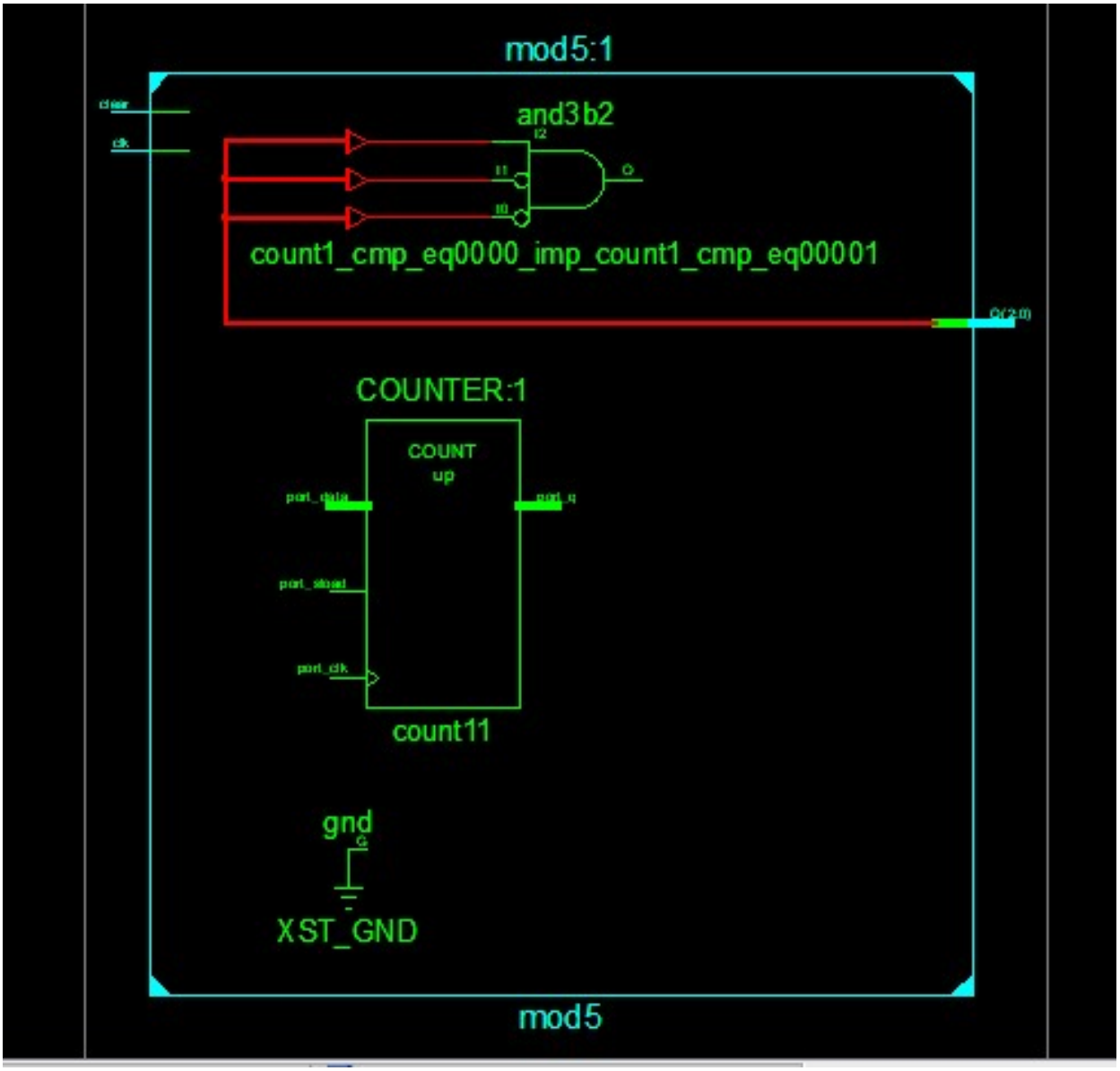
Practical No -: 3

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_arith.ALL;
use IEEE.STD_LOGIC_unsigned.ALL;
entity mod5 is
        Port ( clk : in     STD_LOGIC;
                    clear : in     STD_LOGIC;
                    Q : out     STD_LOGIC_VECTOR (2 downto 0));
end mod5;

architecture Behavioral of mod5 is
SIGNAL count1:STD_LOGIC_VECTOR (2 downto 0);

begin
  PROCESS(clk,clear)
begin
if clear='1' then
count1<="000";
elsif rising_edge(clk)then
if count1="100" then
count1<="000";
else
count1<=count1+1;
```

```
end if;

end if;

end process;

Q<=count1;

end Behavioral;
```





TESTBENCH -:

```
LIBRARY ieee;

USE ieee.std_logic_1164.ALL;


ENTITY mm IS

END mm;
```

```
ARCHITECTURE behavior OF mm IS
        -- Component Declaration for the Unit Under Test (UUT)
        COMPONENT mod5
        PORT(
                clk : IN     std_logic;
                clear : IN     std_logic;
                Q : OUT     std_logic_vector(2 downto 0)
               );
        END COMPONENT;
      --Inputs
      signal clk : std_logic := '0';
      signal clear : std_logic := '0';

         --Outputs
      signal Q : std_logic_vector(2 downto 0);

      -- Clock period definitions
      constant clk_period : time := 10 ns;

BEGIN
         -- Instantiate the Unit Under Test (UUT)
      uut: mod5 PORT MAP (
                  clk => clk,
                  clear => clear,
                  Q => Q);
```

```
	-- Clock process definitions
	clk_process :process
	begin
		clk <= '0';
		wait for clk_period/2;
		clk <= '1';
		wait for clk_period/2;
	end process

	-- Stimulus process
	stim_proc: process
	begin
	    -- hold reset state for 100 ns.
		clear<='1';
	    wait for 100 ns;
		clear<='0';
	    wait for 100 ns;
	    wait for clk_period*10;
	    wait;
	end process;
END;
```

Clock Division -:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_arith.ALL;
use IEEE.STD_LOGIC_unsigned.ALL;
entity mod5 is
        Port ( clk : in     STD_LOGIC;
                    clear : in     STD_LOGIC;
                    Q : out     STD_LOGIC_VECTOR (2 downto 0));
end mod5;
architecture Behavioral of mod5 is
SIGNAL count1:STD_LOGIC_VECTOR (2 downto 0);
```

```
signal count:STD_LOGIC_VECTOR (20 downto 0);
signal clk1:STD_LOGIC;
begin
process(clk)
begin
if clk'event and clk='1' then
count<=count+"0001";
end if;
clk1<=count(20);
end process;
  process(clk1,clear)
begin
if clear='1' then
count1<="000";
elsif rising_edge(clk1)then
if count1="100" then
count1<="000";
else
count1<=count1+1;
end if;
end if;
end process;
Q<=count1;
end Behavioral;
```