8) Write VHDL code for the following:

i) Testbench for a 2 bit comparator

Solution:

```
library IEEE;
use IEEE.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity comparator_tb is
end comparator_tb;

architecture TB of comparator_tb is
  component Comparator
    port ( A: in std_logic_vector (1 downto 0);
           B: in std_logic_vector (1 downto 0);
           less: out std_logic;
           equal: out std_logic;
           greater: out std_logic);
  end component;
  signal A,B: std_logic_vector (1 downto 0):= "00";
  signal less, equal, greater: std_logic;
begin
  uut: comparator portmap( A, B, less, equal, greater);
  process
  begin
  A <= "00";
  B <= "00";
  wait for 10ns;
  A <="00"
  B <="01"
  wait for 10ns;
```

```
A <= "01"
B <= "01"
wait for 10ns;
   end process;
end TB;
```

vi) Jk Flip Flop

```
library IEEE;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity JKFF is
   port (J: in std_logic;
         k: in std_logic;
        clk: in std_logic;
         Q: inout std_logic;
        QN: inout std_logic);
end JKFF

architecture JKA of JKFF is
begin
   process (clk, J, k)
   begin
      if (clk='1' and clk'event) then
      if (J='0' and k='0') then
            Q <= Q;
          QN <= QN;
      elsif (J='0' and k='1') then
            Q <= '1';
          QN <= '0';
```

```
elsif (J='1' and K='0') then
    Q<='0';
    QN<='1';
elsif (J='1' and K='1') then
    Q<= NOT Q;
    QN<= NOT QN;
end if;
end if;
end process;
end JKA;
```

iii) 2 bit shift register:

Solution:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity siso_register is
  port (r_in: in std_logic_vector(1 downto 0);
        rst: in std_logic;
        clk: in std_logic;
        r_out: out std_logic_vector(1 downto 0));
end siso_register;

architecture shift of siso_register is
begin
  process(clk, r_in, rst)
  variable s: std_logic_vector(1 downto 0):="00";
  begin
  if(rst='1') then
    s:=00;
```

```
    elsif (clk = '1' and clk'event) then
      s := (r_in & s(1 downto 0));
      r_out <= s(2);
    end if;
  end process;
end shift;
```

9) Write a VHDL code for 3 bit gray to binary encoder.

Solution:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all

entity grey_to_binary is
    port( binary: out std_logic_vector(2 downto 0);
          grey: in std_logic_vector(2 downto 0));
end grey_to_binary;

architecture G2B of grey_to_binary is
begin
  process
  begin
   case grey is
    when "000" => binary <= "000";
    when "001" => binary <= "001";
    when "011" => binary <= "010";
    when "010" => binary <= "011";
    when "110" => binary <= "100";
    when "111" => binary <= "101";
    when "101" => binary <= "110";
    when "100" => binary <= "111";
   end case;
 end process;
end G2B;
```

11) 3-bit binary to grey converter:

Solution:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity binary_to_grey is
    port(binary: in std_logic_vector(2 downto 0);
         grey: out std_logic_vector(2 downto 0));
end binary_to_grey;

architecture beh of binary_to_grey is
begin
   process(binary)
begin
   case binary is
      when "000" => grey <= "000";
      when "001" => grey <= "001";
      when "010" => grey <= "011";
      when "011" => grey <= "010";
      when "100" => grey <= "110";
      when "101" => grey <= "111";
      when "110" => grey <= "101";
      when "111" => grey <="100";
   end case;
   end process;
end beh;
```

MODULE - II

1) Draw the stick daigram and layout for 2 input NAND Gate in CMOS logic. Use LAMBDA rules.

Solution: $Y = \overline{A \cdot B}$

NMOS: AB → series

PMOS: A'+B' → parallel