---JK Flip-Flop with reset
---the description of JK Flip-Flop is based on functional truth table
---concurrent statement and signal assignment are using in this example
---

library ieee;
use ieee.std_logic_1164.all;
---

text

entity JK_FF is
  port ( clock : in bit;
         J, K : in std_logic;
         Reset : in std_logic;
         Q, Qbar : out std_logic
  );
end JK_FF;
---

architecture behv of JK_FF is
  -- define the useful signals here
  signal state: std_logic;
  signal input: std_logic_vector(1 downto 0);
begin
  -- combine inputs into vector
  input <= J & K;
  process(clock, reset)
  begin
    if (reset='1') then
      state <= '0';
    elsif (rising_edge(clock)) then
      -- compare to the truth table
      case (input) is
        when "11" =>
          state <= not state;
        when "10" =>
          state <= '1';
        when "01" =>
          state <= '0';
        when others =>
          null;
      end case;
    end if;
  end process;
  -- concurrent statements
  Q <= state;
  Qbar <= not state;
end behv;
-- BCD to Seven-Segment Decoder with Enable

library ieee;
use ieee.std_logic_1164.all;

entity seven_seg_decoder is
  port ( Enable : in std_logic;
         BCD : in std_logic_vector(3 downto 0);
         SEG_OUT : out std_logic_vector(7 downto 1) );
end seven_seg_decoder;

architecture behv of seven_seg_decoder is
  -- define the useful signals here
  signal TEMP: std_logic_vector(7 downto 1);
begin
  with BCD select
    TEMP <= "1111110" when "0000",
            "0110000" when "0001",
            "1101101" when "0010",
            "1111001" when "0011",
            "0110011" when "0100",
            "1011010" when "0101",
            "1011111" when "0110",
            "1110000" when "0111",
            "1111111" when "1000",
            "1111011" when "1001",
            "1000000" when others;

  SEG_OUT <= "0000000" when Enable = '1'
                     else TEMP;
end behv;