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 sevenseg is
  Port ( button : in STD_LOGIC;
         RST : in STD_LOGIC;
         CLK : in STD_LOGIC;
         clock_480 : in STD_LOGIC;
         clock_centi : in STD_LOGIC;
         anode : out STD_LOGIC_VECTOR (3 downto 0);
         cathode : out STD_LOGIC_VECTOR (7 downto 0));
end sevenseg;

architecture Behavioral of sevenseg is

signal a, b, x, y : integer:=0; --for the 4 leds on the seven-segment display
signal PS, NS : STD_LOGIC_VECTOR (1 downto 0):="00"; --present state/next state
signal ss1, ss2, en : STD_LOGIC:='0';

begin

  process(CLK)
  begin
    if (rising_edge(CLK)) then
      PS <= NS;
    end if;
  end process;

  adding : process (button, RST, clock_centi, PS, NS, ss1, ss2)
  begin
    if RST = '1' then --if reset is "high" then the clock will display all zeros
      a <= 0;
  end process;

end Behavioral;
b <= 0;
x <= 0;
y <= 0;
else
  if (rising_edge(clock_cnt)) then
    if button = '1' then -- to detect "risingedge" for the button
      ss1 <= '1';
    elsif button = '0' then
      ss1 <= '0';
    end if;
    ss2 <= ss1;
  end if;
  if ss2 = '0' and ss1 = '1' then
    en <= not en;
  end if;
  case (PS) is
    when "11" => -- when the clock is already running
      if en = '1' then
        NS <= "11";
y <= y + 1; -- code to have the stopwatch actually count
      if y = 9 then
        x <= x + 1;
y <= 0;
      if x = 9 then
        b <= b + 1;
x <= 0;
      if b = 9 then
        a <= a + 1;
b <= 0;
      if a = 9 then -- rolls over when it gets to 99.99
        a <= 0;
b <= 0;
x <= 0;
y <= 0;
      end if;
    end if;
    end if;
  end case;
eslif en = '0' then
  NS <= "00";
end if;
when "00" => --when the stopwatch is stopped
  if en = '0' then
    NS <= "00";
    a <= a;
    b <= b;
    x <= x;
    y <= y;
  elsif en = '1' then
    NS <= "11";
  end if;

when others => --should never happen
  NS <= "00";
  a <= 0;
  b <= 0;
  x <= 0;
  y <= 0;
end case;
end if;
end if;
end process;

led : process (clock_480) --tells leds when to turn on
variable digit1, digit2 : unsigned (1 downto 0):="00";

begin
  if (rising_edge(clock_480)) then
    case (digit1) is
      when "00" =>
        anode <= "0111";
      when "01" =>
        anode <= "1011";
      when "10" =>
        anode <= "1101";
      when "11" =>
        anode <= "1110";
      end case;

    case (digit2) is
      when "00" =>
        case (a) is
          when 0 =>


cathode <= "00000011";
when 1 =>
cathode <= "10011111";
when 2 =>
cathode <= "00100101";
when 3 =>
cathode <= "00001101";
when 4 =>
cathode <= "10011001";
when 5 =>
cathode <= "01001001";
when 6 =>
cathode <= "01000001";
when 7 =>
cathode <= "00011111";
when 8 =>
cathode <= "00000001";
when 9 =>
cathode <= "00011001";
when others =>
cathode <= "11111111";
end case;

when "01" =>
case (b) is
  when 0 =>
cathode <= "00000010";
  when 1 =>
cathode <= "10011110";
  when 2 =>
cathode <= "00100100";
  when 3 =>
cathode <= "00001100";
  when 4 =>
cathode <= "10011000";
  when 5 =>
cathode <= "01001000";
  when 6 =>
cathode <= "01000000";
  when 7 =>
cathode <= "00011110";
  when 8 =>
cathode <= "00000000";
when 9 =>
cathode <= "00011000";
when others =>
cathode <= "11111110";
end case;

when "10" =>
case (x) is
  when 0 =>
cathode <= "00000011";
  when 1 =>
cathode <= "10011111";
  when 2 =>
cathode <= "00100101";
  when 3 =>
cathode <= "00001101";
  when 4 =>
cathode <= "10011001";
  when 5 =>
cathode <= "01001001";
  when 6 =>
cathode <= "01000001";
  when 7 =>
cathode <= "00011111";
  when 8 =>
cathode <= "00000001";
  when 9 =>
cathode <= "00011001";
  when others =>
cathode <= "11111111";
end case;

when "11" =>
case (y) is
  when 0 =>
cathode <= "00000011";
  when 1 =>
cathode <= "10011111";
  when 2 =>
cathode <= "00100101";
  when 3 =>
cathode <= "00001101";
  when 4 =>
cathode <= "10011001";
when 5 =>
  cathode <= "01001001";
when 6 =>
  cathode <= "01000001";
when 7 =>
  cathode <= "00011111";
when 8 =>
  cathode <= "00000001";
when 9 =>
  cathode <= "00011001";
when others =>
  cathode <= "11111111";
end case;
end case;
digit1 := digit1 + 1;
digit2 := digit2 + 1;
end if;
end process;

end Behavioral;