```
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 display is
    Port ( refreshClock : in STD_LOGIC;
           clockDig1 : in STD_LOGIC;
           buttonbus : in STD_LOGIC_VECTOR (3 downto 0);
           cathode : out STD_LOGIC_VECTOR (7 downto 0);
           anode : out STD_LOGIC_VECTOR (3 downto 0));
end display;

architecture Behavioral of display is

    signal en, lap, displayer : std_logic:= '0';
    signal push_button_sig_start, push_button_sig_save_lap, push_button_sig1,
push_button_sig2, push_button_sig_display_lap, push_button_sig3, lap_count, lap_display
: integer:=0;
    signal dig1Count, dig2Count, dig3Count, dig4Count, use1, use2, use3, use4 : integer := 0;
    signal dig1Lap, dig2Lap, dig3Lap, dig4Lap, dig1Lap2, dig2Lap2, dig3Lap2, dig4Lap2,
dig1Lap3, dig2Lap3, dig3Lap3, dig4Lap3 : integer := 0;
begin

    display : process (refreshClock, clockDig1, buttonbus, en, lap) is
    begin
        if (buttonbus(1) = '1' and en = '0') then
            use1 <= 0;
            use2 <= 0;
            use3 <= 0;
            use4 <= 0;
            dig1Count <= 0;
            dig2Count <= 0;
            dig3Count <= 0;
            dig4Count <= 0;
            dig1lap <= 0;
            dig2lap <= 0;
            dig3lap <= 0;
            dig4lap <= 0;
    dig1lap2 <= 0;
    dig2lap2 <= 0;
    dig3lap2 <= 0;
    dig4lap2 <= 0;
    dig1lap3 <= 0;
    dig2lap3 <= 0;
    dig3lap3 <= 0;
    dig4lap3 <= 0;
end if;
if (displayer = '1' and en = '0') then
    if (lap_display = 1) then
        use1 <= dig1Lap;
        use2 <= dig2Lap;
        use3 <= dig3Lap;
        use4 <= dig4Lap;
    elsif (lap_display = 2) then
        use1 <= dig1Lap2;
        use2 <= dig2Lap2;
        use3 <= dig3Lap2;
        use4 <= dig4Lap2;
    elsif (lap_display = 3) then
        use1 <= dig1Lap3;
        use2 <= dig2Lap3;
        use3 <= dig3Lap3;
        use4 <= dig4Lap3;
    end if;
end if;
if (rising_edge(clockDig1)) then
    if (en = '1') then
        use1 <= dig1Count;
        use2 <= dig2Count;
        use3 <= dig3Count;
        use4 <= dig4Count;
        if (lap = '1') then
            if (lap_count = 1) then
                 dig1Lap <= dig1Count;
                 dig2Lap <= dig2Count;
                 dig3Lap <= dig3Count;
                 dig4Lap <= dig4Count;
            elsif (lap_count = 2) then
                  dig1Lap2 <= dig1Count;
                  dig2Lap2 <= dig2Count;
                  dig3Lap2 <= dig3Count;
                  dig4Lap2 <= dig4Count;
            elsif (lap_count = 3) then
                dig1Lap3 <= dig1Count;
```

```
               dig2Lap3 <= dig2Count;
               dig3Lap3 <= dig3Count;
               dig4Lap3 <= dig4Count;
            end if;
         end if;
         dig1Count <= dig1Count + 1;
         if (dig1Count = 9) then
            dig1Count <= 0;
            dig2Count <= dig2Count + 1;
            if (dig2Count = 9) then
               dig2Count <= 0;
               dig3Count <= dig3Count + 1;
               if (dig3Count = 9) then
                  dig3Count <= 0;
                  dig4Count <= dig4Count + 1;
                  if (dig4Count = 9) then
                     dig4Count <= 0;
                  end if;
               end if;
            end if;
         end if;
      end if;
   end if;
end process display;

process (refreshClock)
variable digit : unsigned (1 downto 0) := "00";
begin
   if(rising_edge(refreshClock)) then
         case digit is
            when "00" =>
               case (use1) is
                  when 0 =>
                     anode <= "1110";
                     cathode <= "00000011";
                  when 1 =>
                     anode <= "1110";
                     cathode <= "11110011";
                  when 2 =>
                     anode <= "1110";
                     cathode <= "00100101";
                  when 3 =>
                     anode <= "1110";
                     cathode <= "00001101";
                  when 4 =>
                     anode <= "1110";
```

```
                 cathode <= "10011001";
              when 5 =>
                 anode <= "1110";
                 cathode <= "01001001";
              when 6 =>
                 anode <= "1110";
                 cathode <= "11000001";
              when 7 =>
                 anode <= "1110";
                 cathode <= "00011111";
              when 8 =>
                 anode <= "1110";
                 cathode <= "00000001";
              when 9 =>
                 anode <= "1110";
                 cathode <= "00011001";
              when others =>
                 anode <= "1110";
                 cathode <= "00000000";
           end case;
        when "01" =>
           case (use2) is
              when 0 =>
                anode <= "1101";
                 cathode <= "00000011";
              when 1 =>
                anode <= "1101";
                 cathode <= "11110011";
              when 2 =>
                anode <= "1101";
                 cathode <= "00100101";
              when 3 =>
                anode <= "1101";
                 cathode <= "00001101";
              when 4 =>
                anode <= "1101";
                 cathode <= "10011001";
              when 5 =>
                anode <= "1101";
                 cathode <= "01001001";
              when 6 =>
                anode <= "1101";
                 cathode <= "11000001";
              when 7 =>
               anode <= "1101";
                 cathode <= "00011111";
        when 8 =>
          anode <= "1101";
            cathode <= "00000001";
        when 9 =>
          anode <= "1101";
            cathode <= "00011001";
        when others =>
          anode <= "1101";
            cathode <= "00000000";
    end case;
when "10" =>
    case (use3) is
        when 0 =>
          anode <= "1011";
            cathode <= "00000010";
        when 1 =>
          anode <= "1011";
            cathode <= "11110010";
        when 2 =>
          anode <= "1011";
            cathode <= "00100100";
        when 3 =>
          anode <= "1011";
            cathode <= "00001100";
        when 4 =>
          anode <= "1011";
            cathode <= "10011000";
        when 5 =>
          anode <= "1011";
            cathode <= "01001000";
        when 6 =>
          anode <= "1011";
            cathode <= "11000000";
        when 7 =>
          anode <= "1011";
            cathode <= "00011110";
        when 8 =>
          anode <= "1011";
            cathode <= "00000000";
        when 9 =>
          anode <= "1011";
            cathode <= "00011000";
        when others =>
          anode <= "1011";
            cathode <= "00000000";
    end case;
```

```
            when "11" =>
               case (use4) is
                  when 0 =>
                     anode <= "0111";
                     cathode <= "00000011";
                  when 1 =>
                     anode <= "0111";
                     cathode <= "11110011";
                  when 2 =>
                     anode <= "0111";
                     cathode <= "00100101";
                  when 3 =>
                     anode <= "0111";
                     cathode <= "00001101";
                  when 4 =>
                     anode <= "0111";
                     cathode <= "10011001";
                  when 5 =>
                     anode <= "0111";
                     cathode <= "01001001";
                  when 6 =>
                     anode <= "0111";
                     cathode <= "11000001";
                  when 7 =>
                     anode <= "0111";
                     cathode <= "00011111";
                  when 8 =>
                     anode <= "0111";
                     cathode <= "00000001";
                  when 9 =>
                     anode <= "0111";
                     cathode <= "00011001";
                  when others =>
                     anode <= "0111";
                     cathode <= "00000000";
               end case;
         end case;
         digit := digit + 1;
   end if;
end process;

process(clockDig1)
begin
   if (rising_edge(clockDig1)) then
     if (buttonbus(3) = '1') then
        push_button_sig_display_lap <= 1;
```

```
         elsif (buttonbus(3) = '0') then
            push_button_sig_display_lap <= 0;
         end if;
         push_button_sig3 <= push_button_sig_display_lap;
         if (push_button_sig3 = 0 and push_button_sig_display_lap = 1) then
            displayer <= '1';
            lap_display <= lap_display + 1;
            if (lap_display = 3) then
               lap_display <= 1;
            end if;
         elsif (push_button_sig3 = 1 and push_button_sig_display_lap = 0) then
            displayer <= '0';
         end if;

         if (buttonbus(2) = '1') then
            push_button_sig_save_lap <= 1;
         elsif (buttonbus(2) = '0') then
            push_button_sig_save_lap <= 0;
         end if;
         push_button_sig2 <= push_button_sig_save_lap;
         if (push_button_sig2 = 0 and push_button_sig_save_lap = 1) then
            lap <= '1';
            lap_count <= lap_count + 1;
            if (lap_count = 3) then
               lap_count <= 1;
            end if;
         elsif (push_button_sig2 = 1 and push_button_sig_save_lap = 0) then
            lap <= '0';
         end if;
         if (buttonbus(0) = '1') then
            push_button_sig_start <= 1;
         elsif (buttonbus(0) = '0') then
            push_button_sig_start <= 0;
         end if;
         push_button_sig1 <= push_button_sig_start;
         if (push_button_sig1 = 0 and push_button_sig_start = 1) then
            en <= not en;
            lap_display <= 0;
         end if;
         if (buttonbus(1) = '1' and en = '0') then
            lap_count <= 0;
            lap_display <= 0;
         end if;
      end if;
   end process;
end Behavioral;
```