```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity washingmachine_top is
        Port (
                CLOCK_50 : in  std_logic;                   -- 50MHz System Clock
                KEY0     : in  std_logic;                 -- KEY0 is the Start button
                KEY1     : in  std_logic;                 -- KEY0 is the Stop button
                KEY3     : in  std_logic;                 -- KEY3 is the Reset button
                SW0      : in  std_logic;                 -- SW0 Enable Spin Dry
                SW1      : in  std_logic;                 -- SW0 Door open close
                LEDG0    : out std_logic;                   -- start/stop water pump
                LEDG1    : out std_logic;                   -- start/stop soap
                LEDG2    : out std_logic;                   -- rotate drum
                LEDG3    : out std_logic;                   -- open water drain
                LEDR0    : out std_logic;                   -- LED Fill the tank will water and soap
                LEDR1    : out std_logic;                   -- LED Rotate the motor to spin and wash
                LEDR2    : out std_logic;                   -- LED drain the water.
                LEDR3    : out std_logic;                   -- LED fill the tank with water only
                LEDR4    : out std_logic;                   -- LED Rotate the motor to spin and wash
                LEDR5    : out std_logic;                   -- LED Drain the water
                LEDR6    : out std_logic;                   -- LED dry spin
                HEX0     : out STD_LOGIC_VECTOR (6 downto 0); -- g,f,e,d,c,b,a
                HEX1     : out STD_LOGIC_VECTOR (6 downto 0)  -- g,f,e,d,c,b,a
        );
end washingmachine_top;

architecture Behavioral of washingmachine_top is

        component debouncer is
                Port (
                        clk      : in  std_logic;
                        btn      : in  std_logic;
                        sig_rise : out std_logic
                );
        end component;

        component clock1s is
                Port (
                        clk    : in  std_logic;
                        reset  : in  std_logic;
                        clk_1s : out std_logic
                );
        end component;
        component timer is
                Port ( clk : in std_logic;
                        reset   : in  std_logic;
                        en      : in  std_logic;
                        counter : out std_logic_vector(7 downto 0) -- output 8-bit counter
                );
        end component;

        component bin2bcd is
                port (
                        bin  : in  std_logic_vector (7 downto 0);
                        bcd1 : out std_logic_vector (3 downto 0);
                        bcd2 : out std_logic_vector (3 downto 0);
                        bcd3 : out std_logic_vector (3 downto 0)
                );
        end component;


        component ssd is
                Port (
                        digit : in  std_logic_vector (3 downto 0);
                        seg   : out std_logic_vector (6 downto 0) -- g,f,e,d,c,b,a
                );
        end component;


        component controller is
                port(
                        clk          : in  std_logic;
                        reset        : in  std_logic;
                        spin_dry     : in  std_logic;                   -- perform spin dry
                        start_wash   : in  std_logic;                   -- start wash process
                        door_open    : in  std_logic;                   -- Check if door is open or closed
                        counter      : in  std_logic_vector(7 downto 0); -- output 8-bit counter
                        LEDR0        : out std_logic;                   -- LED Fill the tank will water and
soap
                        LEDR1        : out std_logic;                   -- LED Rotate the motor to spin
and wash
                        LEDR2        : out std_logic;                   -- LED drain the water.
                        LEDR3        : out std_logic;                   -- LED fill the tank with water only
                        LEDR4        : out std_logic;                   -- LED Rotate the motor to spin
and wash
                        LEDR5        : out std_logic;                   -- LED Drain the water
```

```
            LEDR6        : out std_logic;                   -- LED dry spin
            water_pump   : out std_logic;                    -- start/stop water pump
            soap         : out std_logic;                   -- start/stop soap
            rotate_drum  : out std_logic;                    -- rotate drum
            drain        : out std_logic;                   -- open water drain
            enable_timer : out std_logic                     -- open water drain
        );
    end component;


    signal clk_1s : std_logic;

    signal counter    : std_logic_vector(7 downto 0);
    signal digit_unit : std_logic_vector(3 downto 0);
    signal digit_tens : std_logic_vector(3 downto 0);

    signal en_timer : std_logic;


begin

    -- 1 second clock generator for counter
    CLOCK_1S : clock1s
        Port map(
            clk    => CLOCK_50,
            reset  => KEY3,
            clk_1s => clk_1s
        );

    -- Count down timer
    COUNT : timer
        Port map(
            clk     => clk_1s,
            reset   => KEY3,
            en      => en_timer,
            counter => counter
        );


    -- Binary to BCD Converter
    BCD : bin2bcd
        Port map(
            bin  => counter,
            bcd1 => digit_unit,
                bcd2 => digit_tens,
                bcd3 => open
        );


-- 7 Segment Unit Digit
SSD_UNIT : ssd
        Port map (
                digit => digit_unit,
                seg   => HEX0 -- g,f,e,d,c,b,a
        );


-- 7 Segment Tens Digit
SSD_TENS : ssd
        Port map (
                digit => digit_tens,
                seg   => HEX1
        );


-- Washing machine main controller
STATEMACHINE : controller
        port map(
                clk          => clk_1s,
                reset        => KEY3,
                spin_dry     => SW0,     -- perform spin dry
                start_wash   => KEY0,    -- start wash process
                door_open    => SW1,     -- Check if door is open or closed
                counter      => counter, -- output 8-bit counter
                LEDR0        => LEDR0,   -- LED Fill the tank will water and soap
                LEDR1        => LEDR1,   -- LED Rotate the motor to spin and wash
                LEDR2        => LEDR2,   -- LED drain the water.
                LEDR3        => LEDR3,   -- LED fill the tank with water only
                LEDR4        => LEDR4,   -- LED Rotate the motor to spin and wash
                LEDR5        => LEDR5,   -- LED Drain the water
                LEDR6        => LEDR6,   -- LED dry spin
                water_pump   => LEDG0,   -- start/stop water pump
                soap         => LEDG1,   -- start/stop soap
                rotate_drum  => LEDG2,   -- rotate drum
                drain        => LEDG3,   -- open water drain
                enable_timer => en_timer -- open water drain
        );
```

```
end Behavioral;




LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
use ieee.numeric_std.all;
entity bin2bcd is
        port (
                bin : in std_logic_vector (7 downto 0);
                bcd1 : out std_logic_vector (3 downto 0);
                bcd2 : out std_logic_vector (3 downto 0);
                bcd3 : out std_logic_vector (3 downto 0)
                );
end entity;

architecture rtl of bin2bcd is
begin
        process ( bin )
                variable binx : std_logic_vector (7 downto 0) ;
                variable bcd  : std_logic_vector (11 downto 0) ;
        begin
                bcd  := (others => '0') ;
                binx := bin(7 downto 0) ;

                for i in binx'range loop
                        if bcd(3 downto 0) > "0100" then
                                bcd(3 downto 0) := std_logic_vector(unsigned( bcd(3 downto 0)) +
"0011");

                        end if ;
                        if bcd(7 downto 4) > "0100" then
                                bcd(7 downto 4) := std_logic_vector(unsigned( bcd(7 downto 4)) +
"0011");
                        end if ;
                        bcd  := bcd(10 downto 0) & binx(7) ;
                        binx := binx(6 downto 0) & '0' ;
                end loop ;

                bcd3 <= bcd(11 downto 8) ;
                bcd2 <= bcd(7 downto 4) ;
        bcd1 <= bcd(3 downto 0) ;
    end process ;
end architecture;



library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity clock1s is
    Port (
        clk     : in  STD_LOGIC;
        reset    : in  STD_LOGIC;
        clk_1s : out STD_LOGIC
    );
end clock1s;

architecture behavioral of clock1s is
    signal temporal : STD_LOGIC;
    signal counter  : integer range 0 to 24999999 := 0; -- period = time * system clock =>
500ms * 50MHz => 50000000/2 => 25000000
begin
    process (reset, clk) begin
        if (reset = '0') then
            temporal <= '0';
            counter  <= 0;
        elsif (clk'event and clk='1') then
            if (counter = 24999999) then
                temporal <= NOT(temporal);
                counter  <= 0;
            else
                counter <= counter + 1;
            end if;
        end if;
    end process;

    clk_1s <= temporal;
end behavioral;

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity ssd is
    Port (
        digit : in  STD_LOGIC_VECTOR (3 downto 0);
        seg   : out STD_LOGIC_VECTOR (6 downto 0) -- g,f,e,d,c,b,a
    );
end ssd;

architecture Behavioral of ssd is
begin
    process(digit)
    begin
        case digit is
            when "0000" => seg <= "1000000"; -- "0"
            when "0001" => seg <= "1111001"; -- "1"
            when "0010" => seg <= "0100100"; -- "2"
            when "0011" => seg <= "0110000"; -- "3"
            when "0100" => seg <= "0011001"; -- "4"
            when "0101" => seg <= "0010010"; -- "5"
            when "0110" => seg <= "0000010"; -- "6"
            when "0111" => seg <= "1111000"; -- "7"
            when "1000" => seg <= "0000000"; -- "8"
            when "1001" => seg <= "0010000"; -- "9"
            when "1010" => seg <= "0000100"; -- A
            when "1011" => seg <= "0000011"; -- b
            when "1100" => seg <= "1000110"; -- C
            when "1101" => seg <= "0100001"; -- d
            when "1110" => seg <= "0000110"; -- E
            when "1111" => seg <= "0001110"; -- F
            when others => seg <= "1111111"; -- F
        end case;
    end process;
end Behavioral;

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity ssd is
    Port (
        digit : in  STD_LOGIC_VECTOR (3 downto 0);
        seg   : out STD_LOGIC_VECTOR (6 downto 0) -- g,f,e,d,c,b,a
    );
end ssd;

architecture Behavioral of ssd is
begin
    process(digit)
    begin
        case digit is
            when "0000" => seg <= "1000000"; -- "0"
            when "0001" => seg <= "1111001"; -- "1"
            when "0010" => seg <= "0100100"; -- "2"
            when "0011" => seg <= "0110000"; -- "3"
            when "0100" => seg <= "0011001"; -- "4"
            when "0101" => seg <= "0010010"; -- "5"
            when "0110" => seg <= "0000010"; -- "6"
            when "0111" => seg <= "1111000"; -- "7"
            when "1000" => seg <= "0000000"; -- "8"
            when "1001" => seg <= "0010000"; -- "9"
            when "1010" => seg <= "0000100"; -- A
            when "1011" => seg <= "0000011"; -- b
            when "1100" => seg <= "1000110"; -- C
            when "1101" => seg <= "0100001"; -- d
            when "1110" => seg <= "0000110"; -- E
            when "1111" => seg <= "0001110"; -- F
            when others => seg <= "1111111"; -- F
        end case;
    end process;
end Behavioral;


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity controller is
    port(
        clk        : in  std_logic;
        reset      : in  std_logic;
        spin_dry   : in  std_logic;                    -- perform spin dry
        start_wash : in  std_logic;                    -- start wash process
        door_open  : in  std_logic;                    -- Check if door is open or closed
        counter    : in  std_logic_vector(7 downto 0); -- output 8-bit counter
        LEDR0      : out std_logic;                    -- LED Fill the tank will water and soap
        LEDR1      : out std_logic;                    -- LED Rotate the motor to spin and wash
        LEDR2      : out std_logic;                    -- LED drain the water.
        LEDR3      : out std_logic;                    -- LED fill the tank with water only
        LEDR4      : out std_logic;                    -- LED Rotate the motor to spin and wash
        LEDR5      : out std_logic;                    -- LED Drain the water
        LEDR6      : out std_logic;                    -- LED dry spin
        water_pump   : out std_logic; -- start/stop water pump
        soap         : out std_logic; -- start/stop soap
        rotate_drum  : out std_logic; -- rotate drum
        drain       : out std_logic; -- open water drain
        enable_timer : out std_logic  -- open water drain
    );
end controller;

architecture Behavioral of controller is
    TYPE state_type IS (zero,one,two,three,four,five,six,seven);
    SIGNAL state : state_type;
begin

    next_state_logic : process(clk)
    begin
        if(clk'event and clk='1') then
            case state is
                when zero => -- Wait for Start
                    if (door_open = '1') then
                        if start_wash = '0' then
                            state <= zero;
                        end if;
                    elsif (door_open = '0') then
                        if (start_wash='1') then
                            state <= one;
                        end if;
                    end if;
                when one =>             -- Fill the tank will water and soap
                    if (counter = x"32") then -- fill the water tank for 10
seconds. Counter 60 to 50
                        state <= two;
                    end if;
                when two =>             -- Rotate the motor to spin and wash
                    if (counter = x"28") then -- rotate the spinner for 10
seconds. counter 50 to 40
                        state <= three;
                    end if;
                when three =>           -- drain the water.
                    if (counter = x"1E") then -- drain the water 10 seconds.
counter 40 to 30
                        state <= four;
                    end if;
                when four =>            -- fill the tank with water only
                    if (counter = x"14") then -- fill the tank with water only 10
seconds. counter 30 to 20
                        state <= five;
                    end if;
                when five =>                -- Rotate the motor to spin and wash
                    if (counter = x"0A") then --  Rotate the motor to spin and
wash for 10 seconds if spin dry is enabled. 20 - 10
                        state <= six;
                    end if;
                when six => -- Drain the water
                    if (spin_dry='0')then
                        if (counter = x"00") then --  open drain for 10
seconds if 10 - 0
                            state <= zero;
                        end if;
                    elsif (spin_dry='1') then
                        state <= seven;
                    end if;
                when seven =>
                    if (counter = x"00") then --  Rotate the motor to dry spin for
10 seconds if spin dry is enabled. 10 - 0
                        state <= zero;      -- reset the state
                    end if;
            end case;

            if (door_open = '1') then -- if someone opens the door, reset the machine
                state <= zero;
            end if;

        end if;
    end process;

    output_logic : process(reset,state,clk)
    begin
        if reset = '1' then
            case state is
                when zero => -- Wait for Start
                    water_pump   <= '0';
                    soap         <= '0';
                    rotate_drum  <= '0';
                    drain        <= '0';
                    enable_timer <= '0';
                    LEDR0        <= '0';
                    LEDR1        <= '0';
                    LEDR2        <= '0';
                    LEDR3        <= '0';
                    LEDR4        <= '0';
                    LEDR5        <= '0';
                    LEDR6       <= '0';
                when one => -- Fill the tank will water and soap
                    water_pump   <= '1';
                    soap        <= '1';
                    rotate_drum  <= '0';
                    drain       <= '0';
                    enable_timer <= '1';
                    LEDR0       <= '1';
                    LEDR1       <= '0';
                    LEDR2       <= '0';
                    LEDR3       <= '0';
                    LEDR4       <= '0';
                    LEDR5       <= '0';
                    LEDR6       <= '0';
                when two => -- Rotate the motor to spin and wash
                    water_pump   <= '0';
                    soap        <= '0';
                    rotate_drum  <= '1';
                    drain       <= '0';
                    enable_timer <= '1';
                    LEDR0       <= '1';
                    LEDR1       <= '1';
                    LEDR2       <= '0';
                    LEDR3       <= '0';
                    LEDR4       <= '0';
                    LEDR5       <= '0';
                    LEDR6       <= '0';
                when three => -- drain the water.
                    water_pump   <= '0';
                    soap        <= '0';
                    rotate_drum  <= '0';
                    drain       <= '1';
                    enable_timer <= '1';
                    LEDR0       <= '1';
                    LEDR1       <= '1';
                    LEDR2       <= '1';
                    LEDR3       <= '0';
                    LEDR4       <= '0';
                    LEDR5       <= '0';
                    LEDR6       <= '0';
                when four => -- fill the tank with water only
                    water_pump   <= '1';
                    soap        <= '0';
                    rotate_drum  <= '0';
                    drain        <= '0';
                    enable_timer <= '1';
                    LEDR0        <= '1';
                    LEDR1        <= '1';
                    LEDR2        <= '1';
                    LEDR3        <= '1';
                    LEDR4        <= '0';
                    LEDR5        <= '0';
                    LEDR6        <= '0';
                when five => -- Rotate the motor to spin and wash
                    water_pump   <= '0';
                    soap         <= '0';
                    rotate_drum  <= '1';
                    drain        <= '0';
                    enable_timer <= '1';
                    LEDR0        <= '1';
                    LEDR1        <= '1';
                    LEDR2        <= '1';
                    LEDR3        <= '1';
                    LEDR4        <= '1';
                    LEDR5        <= '0';
                    LEDR6        <= '0';
                when six => -- Drain the water
                    water_pump   <= '0';
                    soap         <= '0';
                    rotate_drum  <= '0';
                    drain        <= '1';
                    enable_timer <= '1';
                    LEDR0        <= '1';
                    LEDR1        <= '1';
                    LEDR2        <= '1';
                    LEDR3        <= '1';
                    LEDR4        <= '1';
                    LEDR5        <= '1';
                    LEDR6        <= '0';
                when seven => -- spin dry
                    water_pump   <= '0';
                    soap         <= '0';
                    rotate_drum  <= '1';
                    drain        <= '1';
                    enable_timer <= '1';
                    LEDR0        <= '1';
                    LEDR1        <= '1';
                    LEDR2        <= '1';
```

```
                    LEDR3       <= '1';
                    LEDR4       <= '1';
                    LEDR5       <= '1';
                    LEDR6       <= '1';
            end case;
        elsif reset='0' then
            water_pump   <= '0';
            soap         <= '0';
            rotate_drum  <= '0';
            drain        <= '0';
            enable_timer <= '0';
            LEDR0        <= '0';
            LEDR1        <= '0';
            LEDR2        <= '0';
            LEDR3        <= '0';
            LEDR4        <= '0';
            LEDR5        <= '0';
            LEDR6        <= '0';
        end if;
    end process;
end Behavioral;
```