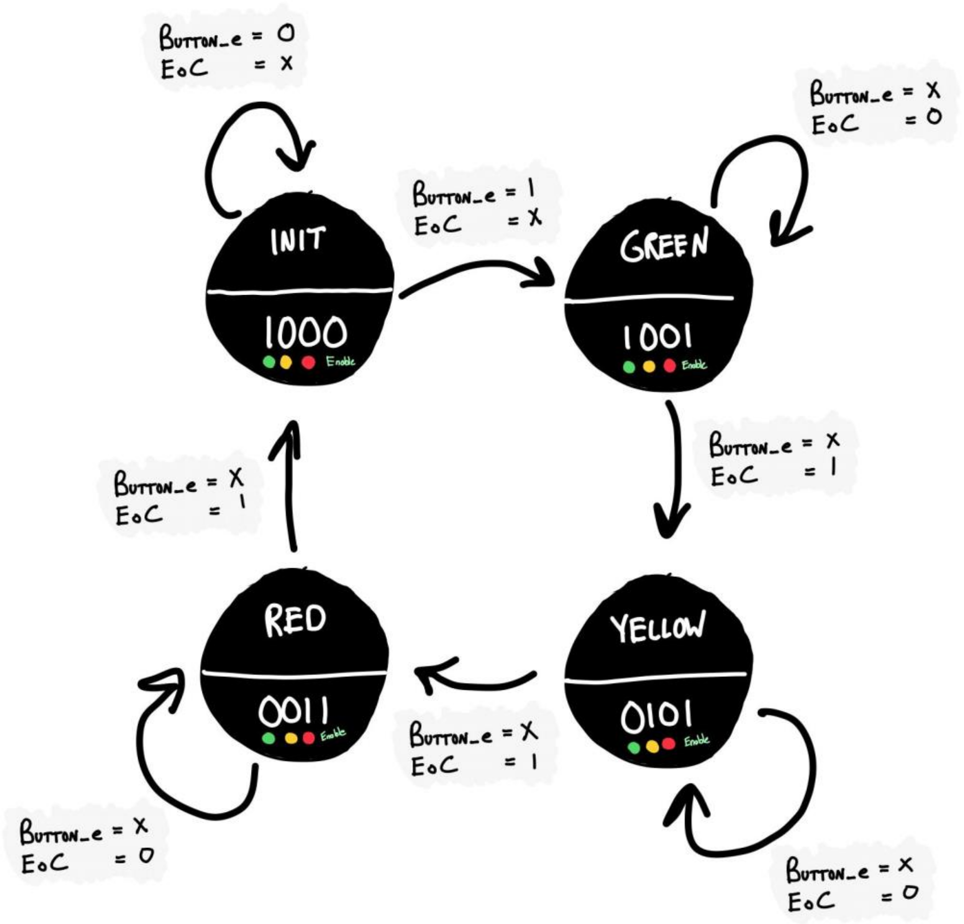
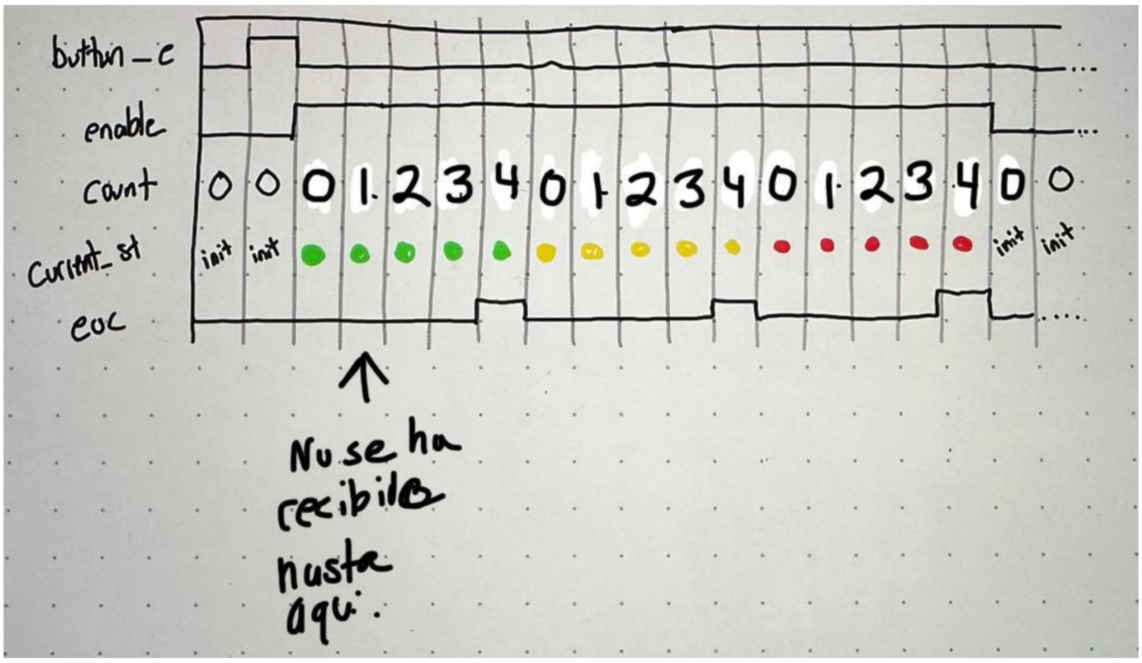
Simon Jones 100504440 - Salvador Ayala 100495832

1. Enter the maximum value of the count to run a 5-second timer with a 1 Hz clock.

The maximum value of the counter is 4 (100 in binary), since the frequency of the clock is 1 tick per second (1Hz). We count from 0 to 4 (takes 5 seconds in total).

2. State diagram of the state machine.





3. VHDL code of the state machine.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY P1 IS
     PORT (
          button, clk, rst : IN std_logic;
          -- ???, TrainerClock, Switch
          r, y, g : OUT std_logic;
          -- LED2, LED1, LED0
          count_out : OUT std_logic_vector(2 DOWNTO 0)
          -- LED7, LED6, LED5
     );
END P1;

-- NOTE: reset is low active!!! it was a typo
ARCHITECTURE behavior OF P1 IS
     SIGNAL q0 : std_logic;
     SIGNAL q1 : std_logic;
     SIGNAL button_e : std_logic;

     TYPE state IS (initial, green, yellow, red);
     SIGNAL current_st, next_st : state;

     SIGNAL enable : std_logic;
     SIGNAL eoc : std_logic;
     SIGNAL count : unsigned(2 DOWNTO 0);
BEGIN
     count_out <= count(2) & count(1) & count(0);

     -- edge detector
     button_e <= q0 AND (NOT q1);
     PROCESS (clk, rst)
     BEGIN
          IF (rst = '0') THEN
               q0 <= '0';
               q1 <= '0';
          ELSIF (clk'EVENT AND clk = '1') THEN
               q0 <= button;
               q1 <= q0;
          END IF;
     END PROCESS;

     -- FSM
     PROCESS (current_st, button_e, eoc) BEGIN
     next_st <= current_st; -- default case for when no
action is performed
     CASE current_st IS
          WHEN initial =>
               IF (button_e = '1') THEN
                    next_st <= green; -- move to green
state
               END IF;
               r <= '0';
               y <= '0';
               g <= '1';
               enable <= '0';
          WHEN green =>
               IF (eoc = '1') THEN
                    next_st <= yellow; -- move to yellow
state
               END IF;
               r <= '0';
               y <= '0';
               g <= '1';
               enable <= '1';
          WHEN yellow =>
               IF (eoc = '1') THEN
                    next_st <= red; -- move to red state
               END IF;
               r <= '0';
               y <= '1';
               g <= '0';
               enable <= '1';
          WHEN red =>
               IF (eoc = '1') THEN
                    next_st <= initial; -- return to
initial state
               END IF;
               r <= '1';
               y <= '0';
               g <= '0';
```

```
                enable <= '1';
            END CASE;
        END PROCESS;

        -- triggers state changes each clock signal
        PROCESS (clk, rst)
        BEGIN
            IF (rst = '0') THEN
                current_st <= initial;
            ELSIF (clk'EVENT AND clk = '1') THEN
                current_st <= next_st;
            END IF;
        END PROCESS;

        -- timer
        -- eoc <= count(2) and not count(1) and not count(0);
        eoc <= '1' WHEN count = "100" ELSE '0';
        PROCESS (clk, rst)
            BEGIN
                IF (rst = '0') THEN
                    count <= "000";
                ELSIF (clk'EVENT AND clk = '1') THEN
                    IF (enable = '0') THEN
                        count <= "000";
                    ELSIF (count = "100") THEN
                        count <= "000";
                    ELSE
                        count <= count + 1;
                    END IF;
                END IF;
            END PROCESS;
    END behavior;
```