```
-----------------------------------------------------------------------------------
-- Example Mealy State machine. SW1 is 'ON' and SW2 is 'OFF'
-- By Colin O'Flynn 2012. Released into the public domain.
-----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity io_connections is
    Port (
             --Reset and Clocks
              RST       : in STD_LOGIC;
              CLK1HZ    : in STD_LOGIC;
              CLK25KHZ  : in STD_LOGIC;

             --Input Switches
              SW1       : in  STD_LOGIC;
              SW2       : in  STD_LOGIC;
              SW3       : in  STD_LOGIC;
              SW4       : in  STD_LOGIC;
              SW5       : in  STD_LOGIC;

             --Output LEDs
              LED1      : out  STD_LOGIC;
              LED2      : out  STD_LOGIC;
              LED3      : out  STD_LOGIC;
              LED4      : out  STD_LOGIC;
              LED5      : out  STD_LOGIC;
              LED6      : out  STD_LOGIC;
              LED7      : out  STD_LOGIC;
              LED8      : out  STD_LOGIC
              );
end io_connections;

architecture Behavioral of io_connections is
    -- Build an enumerated type for the state machine
    type state_type is (sOn, sOff);

    -- Register to hold the current state
    signal state : state_type;

    signal sw_off : STD_LOGIC;
    signal sw_on  : STD_LOGIC;
    signal lamp   : STD_LOGIC;
    signal buzzer : STD_LOGIC;

begin
    --Give signals nicer names
    sw_on <= SW1;
    sw_off <= SW2;
    LED1 <= lamp;
    LED2 <= lamp;
    LED3 <= lamp;
    LED4 <= lamp;

    LED7 <= buzzer;

    --The following chunk of code does the state transitions.
    --It simply transitions between the ON and OFF state depending
    --if one of the inputs is held high
    process (CLK1HZ, RST)
    begin
        if RST = '1' then
            state <= sOff;
        elsif (rising_edge(CLK1HZ)) then
            -- Determine the next state synchronously, based on
            -- the current state and the input
            case state is
                when sOff=>
                    if sw_on = '1' then
                        state <= sOn;
                    else
                        state <= sOff;
                    end if;
                when sOn=>
                    if sw_off = '1' then
                        state <= sOff;
                    else
                        state <= sOn;
                    end if;
            end case;
        end if;
    end process;

    -- Since this is a mealy state machine, we determine the
    -- outputs based on current inputs in addition to state
    -- Here we are doing this asyncronously
    process (state, sw_on, sw_off)
    begin
        case state is
            when sOn=>
                lamp <= '1';
                if sw_off = '1' then
                    buzzer <= '1';
                else
                    buzzer <= '0';
                end if;

            when sOff=>
                lamp <= '0';
                if sw_on = '1' then
                    buzzer <= '1';
                else
                    buzzer <= '0';
                end if;
        end case;
    end process;

end Behavioral;
```