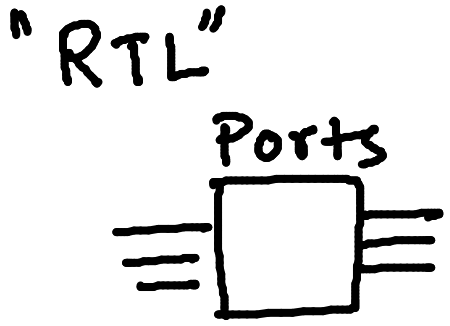
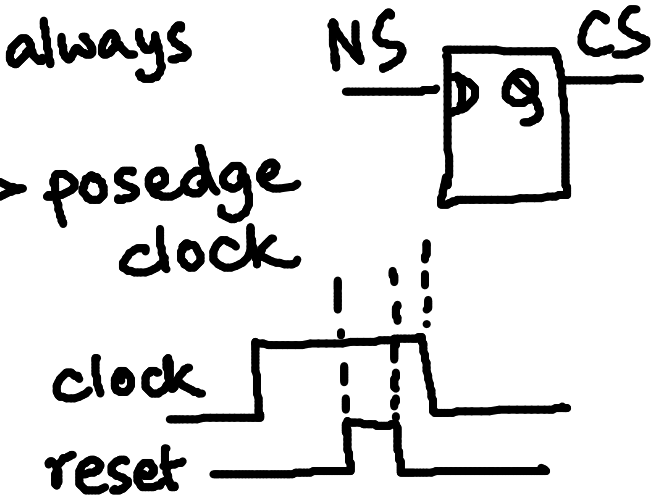
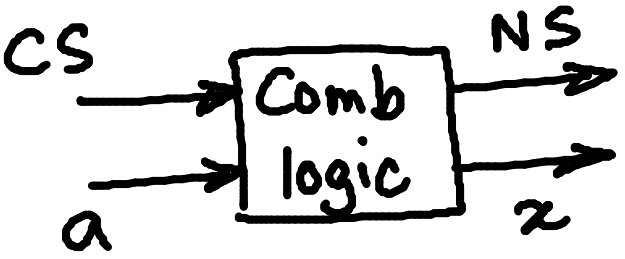
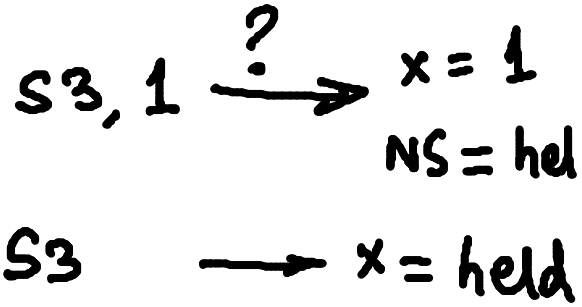
```vhdl
library ieee ;
use ieee.std_logic_1164.all;

-------------------------------------------------------
entity seq_design is
port(   a:              in std_logic;
        clock:          in std_logic;
        reset:          in std_logic;
        x:              out std_logic
);
end seq_design;

-------------------------------------------------------
architecture FSM of seq_design is

    -- define the states of FSM model

    type state_type is (S0, S1, S2, S3);
    signal next_state, current_state: state_type;

begin

    -- cocurrent process#1: state registers
    state_reg: process(clock, reset)
    begin

        if (reset='1') then
            current_state <= S0;
        elsif (clock'event and clock='1') then
            current_state <= next_state;
        end if;

    end process;

    -- cocurrent process#2: combinational logic
    comb_logic: process(current_state, a)
    begin

        -- use case statement to show the
        -- state transistion

        case current_state is

            when S0 =>  x <= '0';
                        if a='0' then
                            next_state <= S0;
                        elsif a ='1' then
                            next_state <= S1;
                        end if;

            when S1 =>  x <= '0';
                        if a='0' then
                            next_state <= S1;
                        elsif a='1' then
                            next_state <= S2;
                        end if;

            when S2 =>  x <= '0';
                        if a='0' then
                            next_state <= S2;
                        elsif a='1' then
                            next_state <= S3;
                        end if;

            when S3 =>  x <= '1';
                        if a='0' then
                            next_state <= S3;
                        elsif a='1' then
                            next_state <= S0;
                        end if;
```









```
            when others =>
                        x <= '0';
                        next_state <= S0;

        end case;

    end process;

end FSM;
```

```
module state_machine (
        output y_out,
        input x_in, clock, reset
);
reg [1:0] current_state, next_state;
parameter S0=2b'00, S1=2b'01, S2=2b'10, S3=2b'11;

always @(posedge clk, negedge reset)
        if (reset == 0) current_state <= S0;
        else current_state <= next_state;

always @(x_in, current_state)
        case(current_state)
        S0: if(x_in) next_state <= S1; else next_state <= S0;
        S1: if(x_in) next_state <= S2; else next_state <= S0;
        S2: if(x_in) next_state <= S3; else next_state <= S0;
        S3: if(x_in) next_state <= S3; else next_state <= S0;
        default: next_state <= S0;
        endcase

always @(current_state)
        case(current_state)
        S0: y_out <= 0;
        S1: y_out <= 0;
        S2: y_out <= 0;
        S3: y_out <= 1;
        endcase
endmodule
```