FSM.vhd

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

entity fsm is
    Port (
        reset  : in  STD_LOGIC;
        input  : in  STD_LOGIC;
        output : out STD_LOGIC;
        clk    : in  STD_LOGIC
    );
end fsm;

architecture Behavioral of fsm is

    -- Define the states
    type state_type is (A, B, C, D);
    signal present_state, next_state : state_type;

begin

    -- State transition process
    process (clk, reset)
    begin
        if (reset = '1') then
            present_state <= A;   -- Default state on reset
        elsif rising_edge(clk) then
            present_state <= next_state;   -- Move to next state
        end if;
    end process;
```

```
-- Next-state and output logic
process (present_state, input)
begin
   case present_state is
      when A =>
         if (input = '0') then
            output <= '1';
            next_state <= C;
         else
            output <= '0';
            next_state <= B;
         end if;

      when B =>
         if (input = '0') then
            output <= '0';
            next_state <= D;
         else
            output <= '1';
            next_state <= B;
         end if;

      when C =>
         if (input = '0') then
            output <= '1';
            next_state <= D;
         else
            output <= '1';
            next_state <= C;
         end if;
```

```
        when D =>
            if (input = '0') then
                output <= '1';
                next_state <= A;
            else
                output <= '0';
                next_state <= D;
            end if;
    end case;
  end process;
end Behavioral;
```

FSM.tb_vhd

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

entity fsm_tb is
end fsm_tb;

architecture Behavioral of fsm_tb is

  -- Component Declaration for the Unit Under Test (UUT)
  component fsm
    Port (
      reset  : in  STD_LOGIC;
      input  : in  STD_LOGIC;
      output : out STD_LOGIC;
      clk    : in  STD_LOGIC
    );
  end component;
```

```
    -- Signals for simulation
    signal clk_tb    : STD_LOGIC := '0';
    signal reset_tb  : STD_LOGIC := '0';
    signal input_tb  : STD_LOGIC := '0';
    signal output_tb : STD_LOGIC;

    -- Optional: to view internal signals (if you edit fsm.vhd to make them visible)
    -- signal present_state, next_state : state_available;

    constant clk_period : time := 10 ns;

begin

    -- Instantiate the Unit Under Test (UUT)
    uut: fsm
        port map (
            reset  => reset_tb,
            input  => input_tb,
            output => output_tb,
            clk    => clk_tb
        );

    -- Clock generation process
    clk_process : process
    begin
        clk_tb <= '0';
        wait for clk_period / 2;
        clk_tb <= '1';
        wait for clk_period / 2;
    end process;
```

```
-- Stimulus process
stim_proc: process
begin
   report "===== FSM Simulation Start =====";

   -- Initial Reset
   reset_tb <= '1';
   wait for 20 ns;
   reset_tb <= '0';
   report "Reset Deasserted, FSM begins operation.";

   -- Apply test pattern
   wait for 10 ns;
   input_tb <= '0';  report "Input set to 0";
   wait for 20 ns;

   input_tb <= '1';  report "Input set to 1";
   wait for 20 ns;

   input_tb <= '0';  report "Input set to 0";
   wait for 20 ns;

   input_tb <= '1';  report "Input set to 1";
   wait for 20 ns;

   input_tb <= '0';  report "Input set to 0";
   wait for 40 ns;

   -- Trigger another reset
   report "Asserting Reset...";
   reset_tb <= '1';
```

```
        wait for 10 ns;
        reset_tb <= '0';
        report "Reset Released.";

        -- Continue with more transitions
        wait for 10 ns;
        input_tb <= '1';  report "Input set to 1";
        wait for 30 ns;

        input_tb <= '0';  report "Input set to 0";
        wait for 30 ns;

        input_tb <= '1';  report "Input set to 1";
        wait for 30 ns;

        -- End simulation
        report "===== FSM Simulation Complete =====";
        wait;
    end process;

end Behavioral;
```