Lab03 – State machines

1 Very simple Moore state machine

```
library ieee;
use ieee.std_logic_1164.all;
entity Moore1 is port(
switch, Clk : in std_logic;
o : out std_logic
);end entity;
architecture arch of Moore1 is
    type statetype is (sON, sOFF);
    signal currentState, nextState : statetype;
begin
    fsm1 : process(switch, currentState)
    begin
        case currentState is
            when sON => o <= '1';
            if switch = '1' then nextState <= sOFF;
            else nextState <= sON; end if;
            when sOFF => o <= '0';
            if switch = '1' then nextState <= sON;
            else nextState <= sOFF; end if;
        end case;
    end process;

    fsm2 : process(Clk)
    begin
    if(Clk'event) and (Clk='1') then
    currentState <= nextState;
    end if;
    end process;
    end arch;
```

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

entity Moore1_tb is
end entity;

architecture ben of Moore1_tb is
    component Moore1 is
    port(
    switch, Clk : in std_logic;
    o : out std_logic
    );end component;

    signal clk : std_logic := '0';
    signal sw, o : std_logic :='0';

begin

    uut : Moore1
    port map(Clk => clk, switch => sw, o => o);

    EN_process : process
    begin
    clk <= not clk;
    wait for 5 ns;
    end process;

    sw_process : process
    begin
    sw <= not sw;
    wait for 30 ns;
    end process;
end ben;
```

2 Moore state machine

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

entity Moore2 is port(
EN, RST, clock : in std_logic;
o : out std_logic_vector(1 downto 0)
);end entity;

architecture arch of Moore2 is
    type statetype is (c0, c1, c2, c3);
    signal currentState, nextState : statetype;
    begin
fsm1 : process(EN, currentState)
    begin
    case currentState is
        when c0 => o <= "00";
        if EN = '1' then nextState <= c1; else nextState <= c0; end if;
        when c1 => o <= "01";
        if EN = '1' then nextState <= c2; else nextState <= c1; end if;
        when c2 => o <= "10";
        if EN = '1' then nextState <= c3; else nextState <= c2; end if;
        when c3 => o <= "11";
        if EN = '1' then nextState <= c0; else nextState <= c3; end if;
    end case;
end process;

fsm2 : process(clock, RST)
    begin
    if(clock'event) and (clock='1') then
        if RST = '1' then
        currentState <= c0;
        else
        currentState <= nextState;
        end if;
    end if;
end process;
end arch;
```

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

entity Moore2_tb is
end entity;

architecture ben of Moore2_tb is
    component Moore2 is
    port(
    EN, RST, clock : in std_logic;
    o : out std_logic_vector(1 downto 0)
    );end component;

    signal EN, RST, clock : std_logic := '0';
    signal o : std_logic_vector(1 downto 0);

begin

    uut : Moore2
    port map(clock => clock, EN => EN, RST => RST, o => o);

    clock_process : process
    begin
    clock <= not clock;
    wait for 2 ns;
    end process;

    EN_process : process
    begin
    EN <= not EN;
    wait for 15 ns;
    end process;

    RST_process : process
    begin
    RST <= not RST;
    wait for 1 ns;
    RST <= not RST;
    wait for 50 ns;
    end process;
end ben;
```

3 More complex state machine

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

entity Moore3 is port(
EN, RST, UP, clock : in std_logic;
o : out std_logic_vector(1 downto 0)
);end entity;

architecture arch of Moore3 is
    type statetype is (c0, c1, c2, c3);
    signal currentState, nextState : statetype;
begin

    fsm1 : process(EN, UP, currentState)
    begin
        case currentState is
        when c0 => o <= "00";
        if EN = '1' then
            if UP = '1' then
            nextState <= c1;
            else
            nextState <= c3;
        end if;
        else nextState <= c0;
        end if;
        when c1 => o <= "01";
        if EN = '1' then
            if UP = '1' then
            nextState <= c2;
            else
            nextState <= c0;
            end if;
        else nextState <= c1;
        end if;
        when c2 => o <= "10";
        if EN = '1' then
            if UP = '1' then
            nextState <= c3;
            else
            nextState <= c1;
            end if;
        else nextState <= c2;
        end if;
        when c3 => o <= "11";
        if EN = '1' then
            if UP = '1' then
            nextState <= c0;
            else
            nextState <= c2;
            end if;
        else nextState <= c3;
        end if;
        end case;
    end process;

    fsm2 : process(clock, RST)
    begin
        if(clock'event) and (clock='1') then
            if RST = '1' then
                currentState <= c0;
            else
                currentState <= nextState;
            end if;
        end if;
    end process;
end arch;
```

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.math_real.all;

entity Moore3_tb is
end entity;

architecture ben of Moore3_tb is
component Moore3 is
    port(
    EN, RST, UP, clock : in std_logic;
    o : out std_logic_vector(1 downto 0)
    );end component;
    signal EN, RST, UP, clock : std_logic := '0';
    signal o : std_logic_vector(1 downto 0);

begin
uut : Moore3
port map(o => o, clock => clock, UP => UP, EN => EN, RST => RST);

clock_process : process
    begin
        clock <= not clock;
        wait for 2 ns;
end process;

    UP_process : process
    begin
        UP <= not UP;
        wait for 50 ns;
    end process;

    EN_process : process
    begin
        EN <= not EN;
        wait for 30 ns;
    end process;

    RST_process : process
    begin
        RST <= not RST;
        wait for 1 ns;
        RST <= not RST;
        wait for 130 ns;
    end process;
end ben;
```