G. Appendix 7 – Finite State Machine Implementations

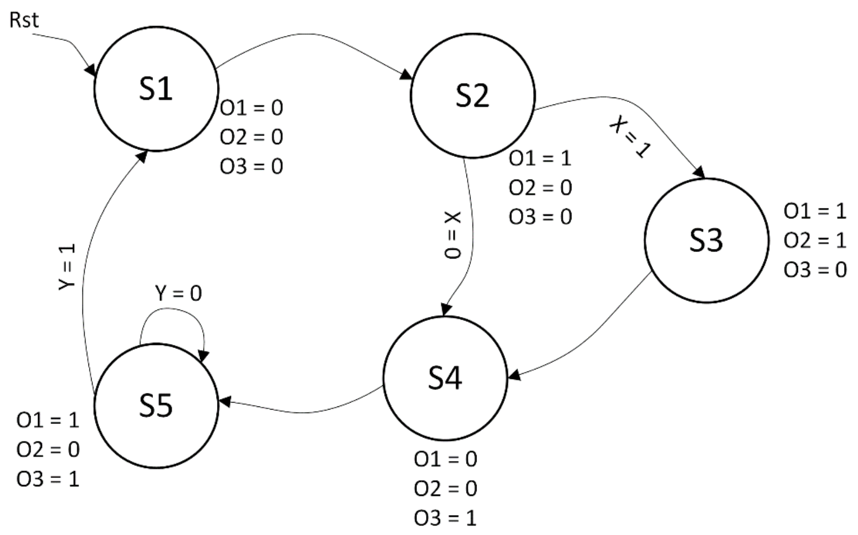

Figure G-1: Finite State Machine Example (XST User Guide)

IO Pins	Description
clk	Positive Edge Clock
Rst	Asynchronous Reset (Active High)
X, Y	FSM Inputs
O1, O2, O3	FSM Outputs

Table G.1: FSM Pin Descriptions

The Xilinx Synthesis technology recognizes Finite State Machines written in VHDL with 1, 2 or 3 processes. A coding example for the Finite State Machine presented in Figure F.1, for each kind of implementation, is given on the next pages. You have to adapt the Finite State Machine implementation to your own FSM description.

VHDL Coding Example: FSM with One Process

```
entity fsm_1 is
        port (
                clk, rst, x, y   : IN std_logic;
                o1, o2, o3       : OUT std_logic
        );
end entity;

architecture beh1 of fsm_1 is
        type state_type is (s1, s2, s3, s4, s5);
        signal state    : state_type;
begin

        process (clk, rst, x, y)
        begin
                if (rst ='1') then
                        state <=s1;
                        o1<='0'; o2<='0'; o3<='0';
                elsif (clk='1' and clk'event) then
                        case state is
                                when s1 =>   state <= s2;
                                             o1<='1'; o2<='0'; o3<='0';
                                when s2 =>   if x = '1' then
                                                     state <= s3;
                                                     o1<='1'; o2<='1'; o3<='0';
                                             else
                                                     state <= s4;
                                                     o1<='0'; o2<='0'; o3<='1';
                                             end if;
                                when s3 =>   state <= s4;
                                             o1<='0'; o2<='0'; o3<='1';
                                when s4 =>   state <= s5;
                                             o1<='1'; o2<='0'; o3<='1';
                                when s5 =>   if y = ‘1’ then
                                                     state <= s1;
                                                     o1<='0'; o2<='0'; o3<='0';
                                             else
                                                     state <= s5;
                                                     o1<='1'; o2<='0'; o3<='1';
                                             end if;
                        end case;
                end if;
        end process;

end beh1;
```

VHDL Coding Example: FSM with Two Processes

```
entity fsm_2 is
        port (
                clk, rst, x, y  : IN std_logic;
                o1, o2, o3      : OUT std_logic
        );
end entity;

architecture beh1 of fsm_2 is
        type state_type is (s1, s2, s3, s4, s5);
        signal state    : state_type;
begin

        process1: process (clk, rst, x, y)
        begin
                if (rst ='1') then
                        state <=s1;
                elsif (clk='1' and clk'event) then
                        case state is
                                when s1 =>      state <= s2;
                                when s2 =>      if x = '1' then
                                                        state <= s3;
                                                else
                                                        state <= s4;
                                                end if;
                                when s3 =>      state <= s4;
                                when s4 =>      state <= s5;
                                when s5 =>      if y = ‘1’ then
                                                        state <= s1;
                                                else
                                                        state <= s5;
                                                end if;
                        end case;
                end if;
        end process process1;


        process2: process (state)
        begin
                case state is
                        when s1 => o1<='0'; o2<='0'; o3<='0';
                        when s2 => o1<='1'; o2<='0'; o3<='0';
                        when s3 => o1<='1'; o2<='1'; o3<='0';
                        when s4 => o1<='1'; o2<='0'; o3<='0';
                        when s5 => o1<='1'; o2<='0'; o3<='1’;
                end case;
        end process process2;
end beh1;
```

VHDL Coding Example: FSM with Three Processes

```
entity fsm_3 is
        port (
                clk, rst, x, y   : IN std_logic;
                o1, o2, o3       : OUT std_logic
        );
end entity;

architecture beh1 of fsm_3 is
        type state_type is (s1, s2, s3, s4, s5);
        signal state, next_state      : state_type;
begin
        process1: process (clk, rst)
        begin
                if (reset ='1') then
                        state <=s1;
                elsif (clk='1' and clk’event) then
                        state <= next_state;
                end if;
        end process process1;


        process2: process (state, x, y)
        begin
                case state is
                        when s1 =>  next_state <= s2;
                        when s2 =>  if x = '1' then
                                            next_state <= s3;
                                    else
                                            next_state <= s4;
                                    end if;
                        when s3 =>  next_state <= s4;
                        when s4 =>  next_state <= s5;
                        when s5 =>  if y = ‘1’ then
                                            next_state <= s1;
                                    else
                                            next_state <= s5;
                                    end if;
                end case;
        end process process2;


        process3: process (state)
        begin
                case state is
```

```
                        when s1 => o1<='0'; o2<='0'; o3<='0';
                        when s2 => o1<='1'; o2<='0'; o3<='0';
                        when s3 => o1<='1'; o2<='1'; o3<='0';
                        when s4 => o1<='1'; o2<='0'; o3<='0';
                        when s5 => o1<='1'; o2<='0'; o3<='1’;
                end case;
        end process process3;
end beh1;
```