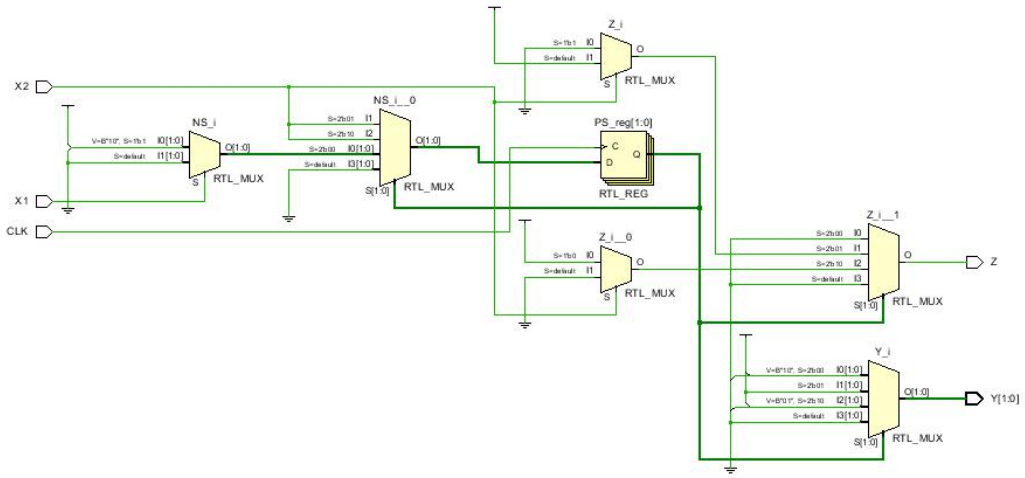
Exercise 2

Code:

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

entity FSM_1 is
   port(X1, X2, CLK: in std_logic;
        Y:     out std_logic_vector(1 downto 0);
       Z:     out std_logic);
end FSM_1;

architecture Behavioral of FSM_1 is
type state_type is (A, B, C);
signal PS, NS: state_type;

begin
sync_process: process(CLK)
   begin

   --update the state at clock rising
   if(CLK' event and CLK = '1') then
      PS <= NS;
   end if;
end process;


comb_process: process(PS, X1, X2)
    begin
    case PS is

       when A =>
          if(X1 = '1') then
             Z <= '0';
             NS <= C;
          elsif(X1 = '0') then
            Z <= '0';
            NS <= A;
          end if;

       when B =>
          if(X2 = '1') then
            Z <= '0';
              NS <= B;
          elsif(X2 = '0') then
             Z <= '1';
             NS <= A;
          end if;

       when C =>
```

```
            if(X2 = '0') then
                Z <= '1';
                NS <= A;
            elsif(X2 = '1') then
              Z <= '0';
                NS <= B;
            end if;

        when others =>
          NS <= A;
          Z <= '0';

    end case;
    end process;

    --update state output
    with PS select
        Y <= "10" when A,
             "11" when B,
             "01" when C,
             "00" when others;

end Behavioral;
```

Schematic:



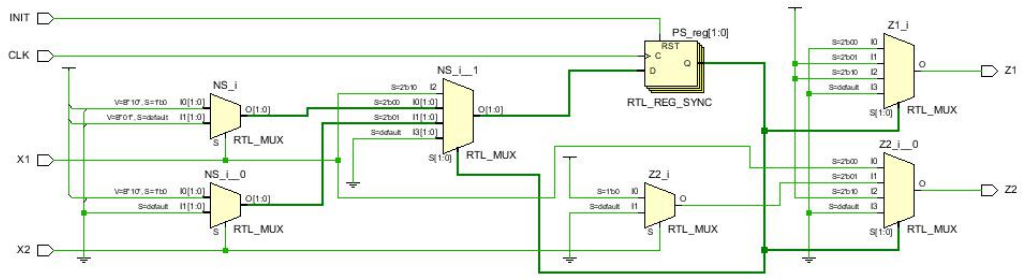
Exercise 4

Code:

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


entity Ch11_4_SFM is
   --Z1 belongs to state (Moore output)
   port(X1, X2, INIT, CLK: in std_logic;
       Z1, Z2:             out std_logic);
end Ch11_4_SFM;

architecture Behavioral of Ch11_4_SFM is
   type state_type is (a, b, c);
   signal PS, NS: state_type;
begin

   sync_process: process(CLK)
   begin
      if(clk' event and clk = '1') then
       if(INIT = '1') then
         PS <= a;
       else PS <= NS;
       end if;
      end if;
   end process;

   comb_process: process(X1, X2)
   begin

      case PS is

         when a =>
            if (X1 = '0') then
               Z2 <= '0';
               NS <= c;
            elsif (X1 = '1') then
               Z2 <= '1';
               NS <= b;
            end if;
         when b =>
            if (X2 = '0') then
               Z2 <= '1';
               NS <= c;
            elsif (X2 = '1') then
               Z2 <= '0';
               NS <= a;
            end if;
         when c =>
```

```
            if (X1 = '0') then
               Z2 <= '1';
               NS <= a;
            elsif (X1 = '1') then
               Z2 <= '1';
               NS <= b;
            end if;

         when others =>
            --default values
            Z2 <= '0'; NS <= a;
      end case;
   end process;

   --update state output
   with PS select
      Z1 <= '0' when a,
            '1' when b,
            '1' when c,
            '0' when others;

end Behavioral;
```

Schematic:



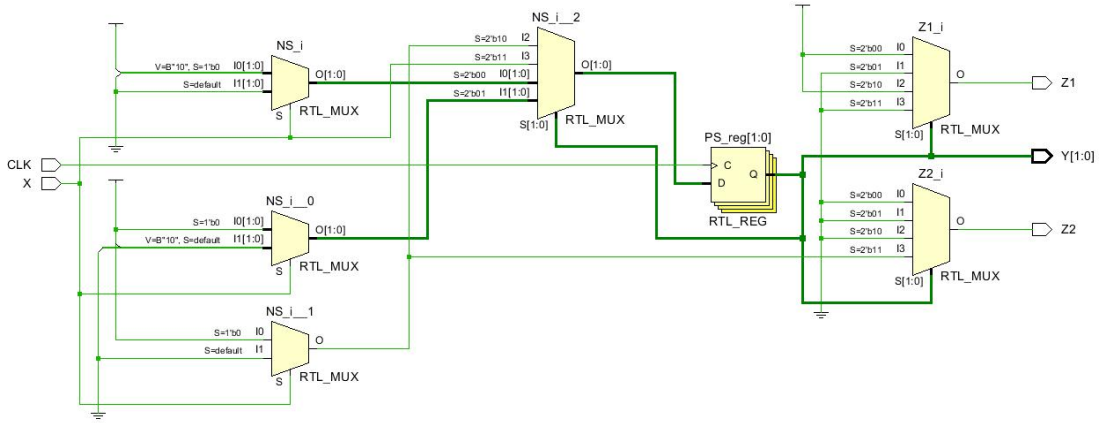
Exercise 6

Code:

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

entity Ch11_6_SFM is
   port (X, CLK:   in std_logic;
         Z1, Z2: out std_logic;
         Y    : out std_logic_vector(1 downto 0));
end Ch11_6_SFM;

architecture Behavioral of Ch11_6_SFM is
   type state_type is (S0, S1, S2, S3);
   signal PS, NS: state_type;
begin

--update state on clock rising
sync_proc: process(CLK)
   begin
   if(CLK' event and CLK = '1') then
      PS <= NS;
   end if;
   end process;

comb_proc: process(X)
   begin
   case PS is

      when S0 =>
         if(X = '0') then Z2 <= '0'; NS <= S2; Z1 <= '1';
         elsif(X = '1') then Z2 <= '0'; NS <= S0; Z1 <= '1';
         end if;

      when S1 =>
         if(X = '0') then Z2 <= '0'; NS <= S3; Z1 <= '0';
         elsif(X = '1') then Z2 <= '0'; NS <= S2; Z1 <= '0';
         end if;

      when S2 =>
         if(X = '0') then Z2 <= '0'; NS <= S1; Z1 <= '1';
         elsif(X = '1') then Z2 <= '0'; NS <= S0; Z1 <= '1';
         end if;

      when S3 =>
         if(X = '0') then Z2 <= '1'; NS <= S0; Z1 <= '0';
         elsif(X = '1') then Z2 <= '0'; NS <= S1; Z1 <= '0';
         end if;
```

```
      when others =>
         Z1 <= '1'; Z2 <= '0'; NS <= S0;
   end case;
   end process;

with PS select
   Y <= "00" when S0,
        "01" when S1,
        "10" when S2,
        "11" when S3,
        "00" when others;

end Behavioral;
```

Schematic:



Exercise 13

Code:

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


entity Ch11_13_SFM is
   port( X1, X2, CLK : in std_logic;
        CS, RD : out std_logic;
        Y :     out std_logic_vector(2 downto 0));
end Ch11_13_SFM;

architecture Behavioral of Ch11_13_SFM is
   type state_type is (a, b, c);
   signal PS, NS: state_type;
begin
--update state on clock rising
sync_proc: process(CLK)
   begin
   if(rising_edge(CLK)) then
      PS <= NS;
   end if;
   end process;

comb_proc: process(PS, X1, X2)
   begin
   case PS is
   when a =>
      if (X1 = '0') then
         CS <= '0'; RD <= '1';
         NS <= b;
      elsif(X1 = '1') then
         CS <= '1'; RD <= '0';
         NS <= c;
      end if;
   when b =>
      CS <= '1'; RD <= '1';
      NS <= c;
   when c =>
      if(X2 = '0') then
         CS <= '0'; RD <= '0';
         NS <= a;
      elsif(X2 = '1') then
         CS <= '0'; RD <= '1';
         NS <= c;
      end if;
    when others =>
        CS <= '0'; RD <= '1'; NS <= a;
   end case;
```

```
   end process;

with PS select
   Y <= "001" when a,
        "010" when b,
        "100" when c,
        "000" when others;

end Behavioral;
```

Schematic: