```
--Anjali Vathanakumaran and Mahi Joshi
--Lab 4: Moore Machine
-- Group 22

library ieee; --moore
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

-- Controls the states of the traffic lights in the NS and EW direction based on pedstrian inputs.
Entity State_Machine_Moore IS Port
(
 clk_input, enable, reset: IN std_logic;
 NSrequest, EWrequest : IN std_logic; -- 1 bit memory to hold the pedestrian request signal in the
NS and EW direction
 flashing, green, amber, red  : OUT std_logic; --the different coloured signals in the NS
direction; flashing = blinking signal
flashingE, greenE, amberE, redE : out std_logic; -- the different coloured signals in the EW
direction; flashingE = blinking signal
 NSclear, EWclear: out std_logic; -- clears the pedestrian request signal at the appropriate states
 NScrossing, EWcrossing : out std_logic; -- controls the pedestrian crossing display signal which
occurs at solid green when it is safe to cross
 stateCounter : out std_logic_vector(3 downto 0) -- counts the current state it is in in binary
 );
END ENTITY;


 Architecture SM of State_Machine_Moore is

 -- the 16 different states of the traffic signals
 TYPE STATE_NAMES IS (S0, S1, S2, S3, S4, S5, S6, S7,S8, S9, S10, S11, S12, S13, S14, S15);   -- 16
STATES


 SIGNAL current_state, next_state   :  STATE_NAMES;      -- signals of type STATE_NAMES


 BEGIN

 --------------------------------------------------------------------------------
 --State Machine: Moore Machine
 --------------------------------------------------------------------------------

Register_Section: PROCESS (clk_input)  -- this process updates with a clock
BEGIN
   IF(rising_edge(clk_input)) THEN
      IF (reset = '1') THEN
      -- if reset is pressed it returns to state 0 which is the first state
         current_state <= S0;
      ELSIF (reset = '0' AND enable = '1') THEN
      -- if reset is not pressed it moves states like regular
         current_state <= next_State;
      END IF;
   END IF;
END PROCESS;



-- TRANSITION LOGIC PROCESS EXAMPLE

Transition_Section: PROCESS (current_state)

BEGIN
  CASE current_state IS
         WHEN S0 =>
            if(EWrequest = '1' AND NSrequest = '0') then
            --if a pedestrian request is made in the Ew direction at S0 and no previous NS request
was made then it jumps to S6
            --Shortens the waiting time for the pedestrians
               next_state <= S6;
            else
            -- if no request is made proceed to next state like regular
               next_state <= S1;
         end if;

         WHEN S1 =>
            if(EWrequest = '1' AND NSrequest = '0') then
            -- if a pedestrian request was made in the EW direction and no previous NS request was
made then jump to S6
               next_state <= S6;
            else
               next_state <= S2;
            end if;
         -- change traffic colours like regular ( flashing -> green -> amber -> red)
         WHEN S2 =>
               next_state <= S3;

         WHEN S3 =>
               next_state <= S4;

         WHEN S4 =>
               next_state <= S5;

         WHEN S5 =>
               next_state <= S6;

         WHEN S6 =>
               next_state <= S7;

         WHEN S7 =>
               next_state <= S8;

         WHEN S8=>
         -- if a pedestrian request is made in the NS direction and no previous EW pedestrian
request (nobody is crossing in the EW direction)
         -- was made then jump to state 14
         -- shortens the waiting time for the pedestrian
            if(NSrequest = '1' AND EWrequest = '0') then
               next_state <= S14;
            else
            --if no request was made proceed to next state like regular
               next_state <= S9;
            end if;

         WHEN S9 =>
            if(NSrequest = '1' AND EWrequest = '0') then
               next_state <= S14;
            else
               next_state <= S10;
            end if;

         WHEN S10 =>
               next_state <= S11;
         WHEN S11 =>
               next_state <= S12;
         WHEN S12 =>
               next_state <= S13;
         WHEN S13 =>
               next_state <= S14;
         WHEN S14 =>
               next_state <= S15;
         WHEN S15 =>
               next_state <= S0;

     END CASE;
 END PROCESS;


-- DECODER SECTION PROCESS

Decoder_Section: PROCESS (current_state)
BEGIN

   NSclear <= '0'; -- set clear signals to 0
   EWclear <= '0';
     CASE current_state IS

         WHEN S0 =>
         -- advanced green which means the flashing signal is on in the NS
         -- red signal is on in the EW
         flashing <= '1';
         green <= '0';
         amber <= '0';
```

```
        red <= '0';
        NSclear <= '0';

        flashingE <= '0';
        greenE <= '0';
        amberE <= '0';
        redE <= '1';
        EWclear <= '0';

        NScrossing <= '0';
        EWcrossing <= '0';

        -- State 0
        stateCounter <= "0000";

        WHEN S1 =>
        -- flashing green is on in NS
        -- red signal is on in the EW
        flashing <= '1';
        green <= '0';
        amber <= '0';
        red <= '0';
        NSclear <= '0';

        flashingE <= '0';
        greenE <= '0';
        amberE <= '0';
        redE <= '1';

        NScrossing <= '0';
        EWcrossing <= '0';
        EWclear <= '0';

        -- State 1
        stateCounter <= "0001";

        WHEN S2 =>
        -- solid green is o in NS
        -- red signal is on in EW
        flashing <= '0';
        green <= '1';
        amber <= '0';
        red <= '0';
        NSclear <= '0';

        flashingE <= '0';
        greenE <= '0';
        amberE <= '0';
        redE <= '1';

        -- solid green is on in Ns, it is safe to cross so NS crossing display is on
        NScrossing <= '1';
        EWcrossing <= '0';
        EWclear <= '0';

        -- State 2
        stateCounter <= "0010";

        WHEN S3 =>
        -- solid green is on in NS
        -- red signal is on in EW
        flashing <= '0';
        green <= '1';
        amber <= '0';
        red <= '0';
        NSclear <= '0';

        flashingE <= '0';
        greenE <= '0';
        amberE <= '0';
        redE <= '1';

        -- safe to cross in NS direction, crossing display is on
        NScrossing <= '1';
        EWcrossing <= '0';
        EWclear <= '0';
        -- State 3
        stateCounter <= "0011";

        WHEN S4 =>
        -- solid green is on is NS
        -- red signal is on is EW
        flashing <= '0';
        green <= '1';
        amber <= '0';
        red <= '0';
        NSclear <= '0';

        flashingE <= '0';
        greenE <= '0';
        amberE <= '0';
        redE <= '1';

        -- solid green in NS, safe to cross so crossing display is on
        NScrossing <= '1';
        EWcrossing <= '0';

        EWclear <= '0';

        -- State 4
        stateCounter <= "0100";

        WHEN S5 =>
        -- solid green is on is NS
        flashing <= '0';
        green <= '1';
        amber <= '0';
        red <= '0';
        NSclear <= '0';

        -- red signal is on in EW
        flashingE <= '0';
        greenE <= '0';
        amberE <= '0';
        redE <= '1';

        --solid green in NS, safe to cross therefore crossing display
        NScrossing <= '1';
        EWcrossing <= '0';

        EWclear <= '0';

        -- State 5
        stateCounter <= "0101";

        WHEN S6 =>
        -- amber is on in NS
        -- red signal is on in EW
        flashing <= '0';
        green <= '0';
        amber <= '1';
        red <= '0';
        -- no longer safe to cross so request is cleared
        NSclear <= '1';

        flashingE <= '0';
        greenE <= '0';
        amberE <= '0';
        redE <= '1';

        -- no longer safe to cross so the crossing display is off
        NScrossing <= '0';
        EWcrossing <= '0';

        EWclear <= '0';

        -- State 6
        stateCounter <= "0110";


        WHEN S7 =>
        -- amber is on in NS
        -- red signal is on in EW
        flashing <= '0';
        green <= '0';
        amber <= '1';
        red <= '0';
        NSclear <= '0';

        flashingE <= '0';
        greenE <= '0';
        amberE <= '0';
        redE <= '1';

        NScrossing <= '0';
        EWcrossing <= '0';

        EWclear <= '0';

        -- State 7
        stateCounter <= "0111";

        WHEN S8 =>
        -- amber is on in NS
        -- flashing signal is on in EW
        flashing <= '0';
        green <= '0';
        amber <= '0';
        red <= '1';
        NSclear <= '0';

        flashingE <= '1';
        greenE <= '0';
        amberE <= '0';
        redE <= '0';

        NScrossing <= '0';
        EWcrossing <= '0';

        EWclear <= '0';

        -- State 8
        stateCounter <= "1000";

        WHEN S9 =>
        -- red signal is on in Ns
        -- flashing signal is on in EW

        flashing <= '0';
        green <= '0';
        amber <= '0';
        red <= '1';
        NSclear <= '0';

        flashingE <= '1';
        greenE <= '0';
        amberE <= '0';
        redE <= '0';

        NScrossing <= '0';
        EWcrossing <= '0';

        EWclear <= '0';

        -- State 9
        stateCounter <= "1001";

        WHEN S10 =>
        -- red signal is on in NS
        -- solid green signal is on in EW
        flashing <= '0';
        green <= '0';
        amber <= '0';
        red <= '1';
        NSclear <= '0';

        flashingE <= '0';
```

```vhdl
		greenE <= '1';
		amberE <= '0';
		redE <= '0';

		-- solid green in EW, safe to cross so the corssing display is on in EW
		NScrossing <= '0';
		EWcrossing <= '1';

		EWclear <= '0';

		--State 10
		stateCounter <= "1010";

		WHEN S11 =>
		-- red signal is on in NS
		-- solid green is on in EW
		flashing <= '0';
		green <= '0';
		amber <= '0';
		red <= '1';
		NSclear <= '0';

		flashingE <= '0';
		greenE <= '1';
		amberE <= '0';
		redE <= '0';

		-- solid green is on, safe to cross so the crossing display is on in Ew
		NScrossing <= '0';
		EWcrossing <= '1';

		EWclear <= '0';

		-- State 11
		stateCounter <= "1011";

		WHEN S12 =>
		-- red signal is on Ns
		-- solid green is on in EW
		flashing <= '0';
		green <= '0';
		amber <= '0';
		red <= '1';
		NSclear <= '0';

		flashingE <= '0';
		greenE <= '1';
		amberE <= '0';
		redE <= '0';

		-- solid green is on, safe to cross so crossing display is on in EW
		NScrossing <= '0';
		EWcrossing <= '1';

		EWclear <= '0';

		-- State 12
		stateCounter <= "1100";

		WHEN S13 =>
		-- red signal is on in NS
		-- solid green signal is on in EW
		flashing <= '0';
		green <= '0';
		amber <= '0';
		red <= '1';
		NSclear <= '0';

		flashingE <= '0';
		greenE <= '1';
		amberE <= '0';
		redE <= '0';

		-- solid green is on, safe to cross so crossing display is on in EW

		NScrossing <= '0';
```

```
        EWcrossing <= '1';

        EWclear <= '0';

        --State 13
        stateCounter <= "1101";

        WHEN S14 =>
        -- red signal is on in NS
        --amber signal is on in Ew
        flashing <= '0';
        green <= '0';
        amber <= '0';
        red <= '1';
        NSclear <= '0';

        flashingE <= '0';
        greenE <= '0';
        amberE <= '1';
        redE <= '0';

        NScrossing <= '0';
        EWcrossing <= '0';

        -- no longer safe to cross so crossing display is off and request is cleared
        EWclear <= '1';

        -- State 14
        stateCounter <= "1110";

        WHEN S15 =>
        -- red signal is on in NS
        -- amber signal is on in EW
        flashing <= '0';
        green <= '0';
        amber <= '0';
        red <= '1';
        NSclear <= '0';

        flashingE <= '0';
        greenE <= '0';
        amberE <= '1';
        redE <= '0';

        NScrossing <= '0';
        EWcrossing <= '0';

        EWclear <= '0';

        --State 15
        stateCounter <= "1111";

        WHEN OTHERS =>
        -- set all vlaues to zero for any other state that is not specified

        flashing <= '0';
        green <= '0';
        amber <= '0';
        red <= '0';
        NSclear <= '0';

        flashingE <= '0';
        greenE <= '0';
        amberE <= '0';
        redE <= '0';

        NScrossing <= '0';
        EWcrossing <= '0';

        EWclear <= '0';



END CASE;
END PROCESS;
END ARCHITECTURE SM;
```