```
1   ------------------------------------------------------------------------
    ----
2   --
3   -- Title       : send_pos_edge_det
4   -- Design      : task1_2
5   -- Author      : Ishabul Haque and Ken Ejinkonye
6   -- Company     : Stony Brook
7   --
8   -- Description : A positive edge detector is used to detect the
9   -- positive edge of the send and to generate a narrow pulse that will
10  -- be used by the spi_tx_shifter to determine when it should start to send
11  -- a byte.
12  -- List of Circuit Features To Be Verified:
13  -- Inputs: rst_bar, clk, send
14  -- Outputs: send_en
15
16
17  ------------------------------------------------------------------------
    ----
18
19  library IEEE;
20  use IEEE.std_logic_1164.all;
21
22  entity send_pos_edge_det is
23      port(
24      rst_bar : in std_logic;     -- asynnchronous system reset
25      clk : in std_logic;         -- system clock
26      send : in std_logic;        -- debounced send input
27      send_en : out std_logic     -- send enable output pulse
28      );
29  end send_pos_edge_det;
30
31
32  architecture moore_fsm of send_pos_edge_det is
33      type state is (state_a, state_b, state_c);
34      signal present_state, next_state : state;
35
36      begin
37          state_reg: process (clk,rst_bar)
38          begin
39              if rst_bar = '0' then
40              -- If rst_bar is enabled, then present state will switch
41              -- to state a
42                  present_state <= state_a;
43              elsif rising_edge(clk) then
44              -- If rst_bar is not enabled, the present state will switch
45              -- to the next state on the rising edge of clock
46                  present_state <= next_state;
47              end if;
48          end process;
49
50          outputs: process (present_state)
51          begin
52              case present_state is
53                  -- Output send_en will only be high when the present state
54                  -- is in state c, otherwise it will output low
55                  when state_c => send_en <= '1';
```

```
56                  when others => send_en <= '0';
57              end case;
58          end process;
59
60          nxt_state: process (present_state, send)
61          begin
62              -- Moore FSM, state values are determined by
63              -- state diagram
64              case present_state is
65                  when state_a =>
66                  if send = '0' then
67                      next_state <= state_b;
68                  else
69                      next_state <= state_b;
70                  end if;
71
72                  when state_b =>
73                  if send = '1' then
74                      next_state <= state_c;
75                  else
76                      next_state <= state_b;
77                  end if;
78
79                  when others =>
80                  if send = '0' then
81                      next_state <= state_b;
82                  else
83                      next_state <= state_a;
84                  end if;
85              end case;
86          end process;
87      end moore_fsm;
88
89
90
```