

```
1 module spi_slave #(parameter transmit_width = 16) (
2     input wire sys_clock,
3     output wire spi_miso,
4     input wire spi_sck,
5     input wire spi_cs,
6     input wire [transmit_width - 1:0] outgoing_data
7     );
8
9 reg [transmit_width - 1:0] out_shift_reg;
10 reg spi_cs_r, spi_cs_rr;
11 reg spi_ck_r, spi_ck_rr;
12 wire spi_cs_detected;
13 wire spi_ck_detected;
14
15 // first double buffer the incoming clock & chip select in order
16 //   to perform edge detection on signals that are crossing a
17 //   clock domain
18
19 always@(posedge sys_clock)
20 begin
21     spi_cs_r <= #1 spi_cs;
22     spi_cs_rr <= #1 spi_cs_r;
23     spi_ck_r <= #1 spi_sck;
24     spi_ck_rr <= #1 spi_ck_r;
25 end
26
27 //chip select samples on low going transition while
28 //   clock samples on high going transition...
29
30 assign #1 spi_cs_detected = (spi_cs_rr && ~spi_cs_r);
31 assign #1 spi_ck_detected = (~spi_ck_rr && spi_ck_r);
32
33 always @(posedge sys_clock)
34 begin
35     if(spi_cs_detected)
36         out_shift_reg[transmit_width - 1:0] <= #1 outgoing_data[transmit_width - 1:0];
37     else if(spi_ck_detected)
38         out_shift_reg[transmit_width - 1:0] <= #1 {0,out_shift_reg[transmit_width - 1:1]};
39 end
40
41 assign #1 spi_miso = out_shift_reg[0];
42
43 endmodule
```