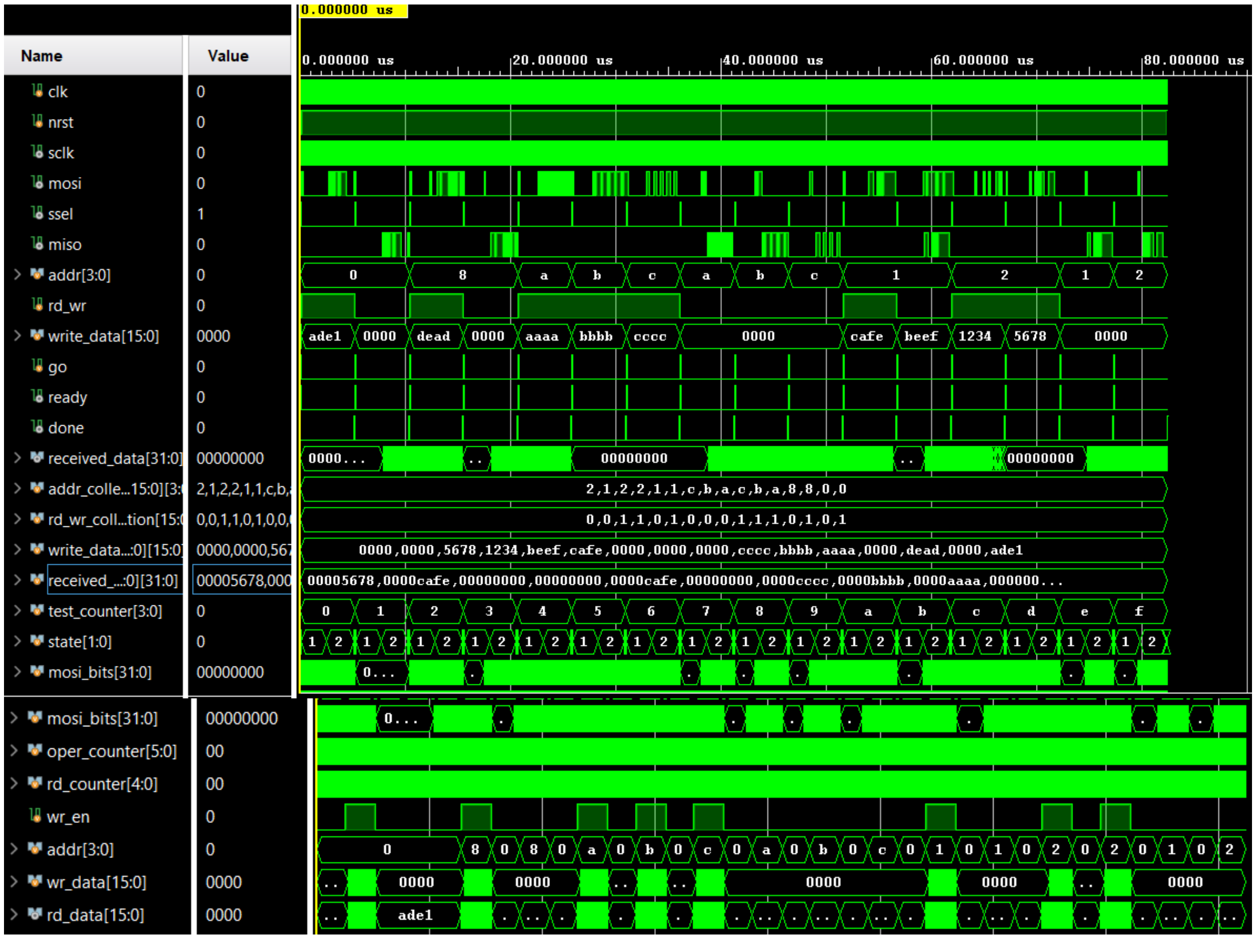
John Danielle T Castor
20[redacted]
11-24-23
Project 4

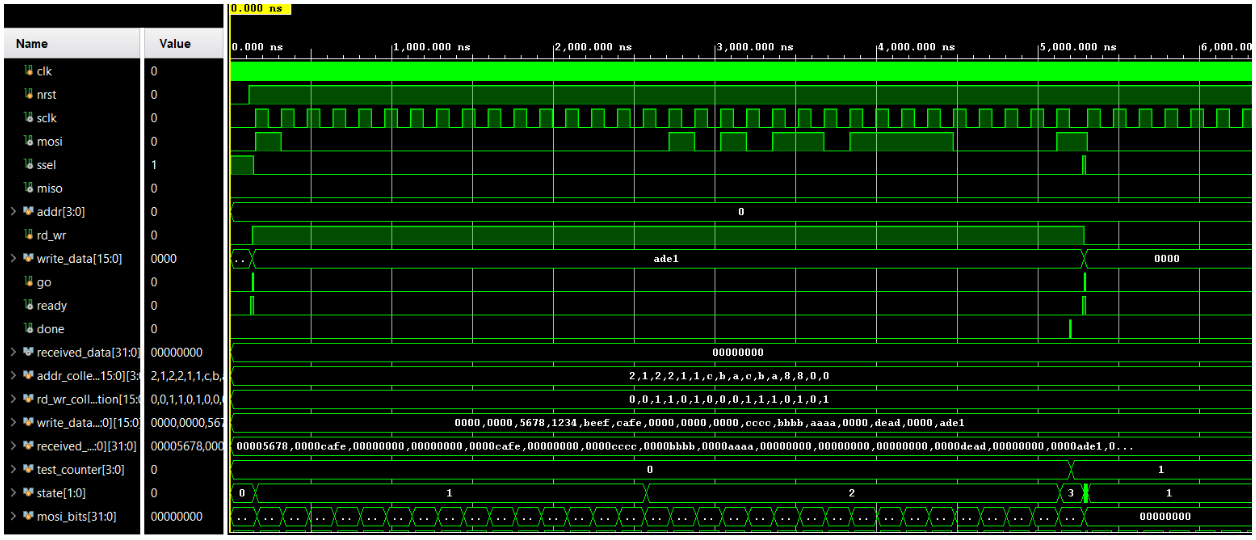
Behavioral Simulation Waveforms and Console



Name	Value
clk	0
nrst	0
sclk	0
mosi	0
ssel	1
miso	0
addr[3:0]	0
rd_wr	0
write_data[15:0]	0000
go	0
ready	0
done	0
received_data[31:0]	00000000
addr_colle...15:0][3:	2,1,2,2,1,1,c,b,
rd_wr_coll...tion[15:	0,0,1,1,0,1,0,0,
write_data...0][15:0]	0000,0000,567
received_...:0][31:0]	00005678,000
test_counter[3:0]	0
state[1:0]	0
mosi_bits[31:0]	00000000

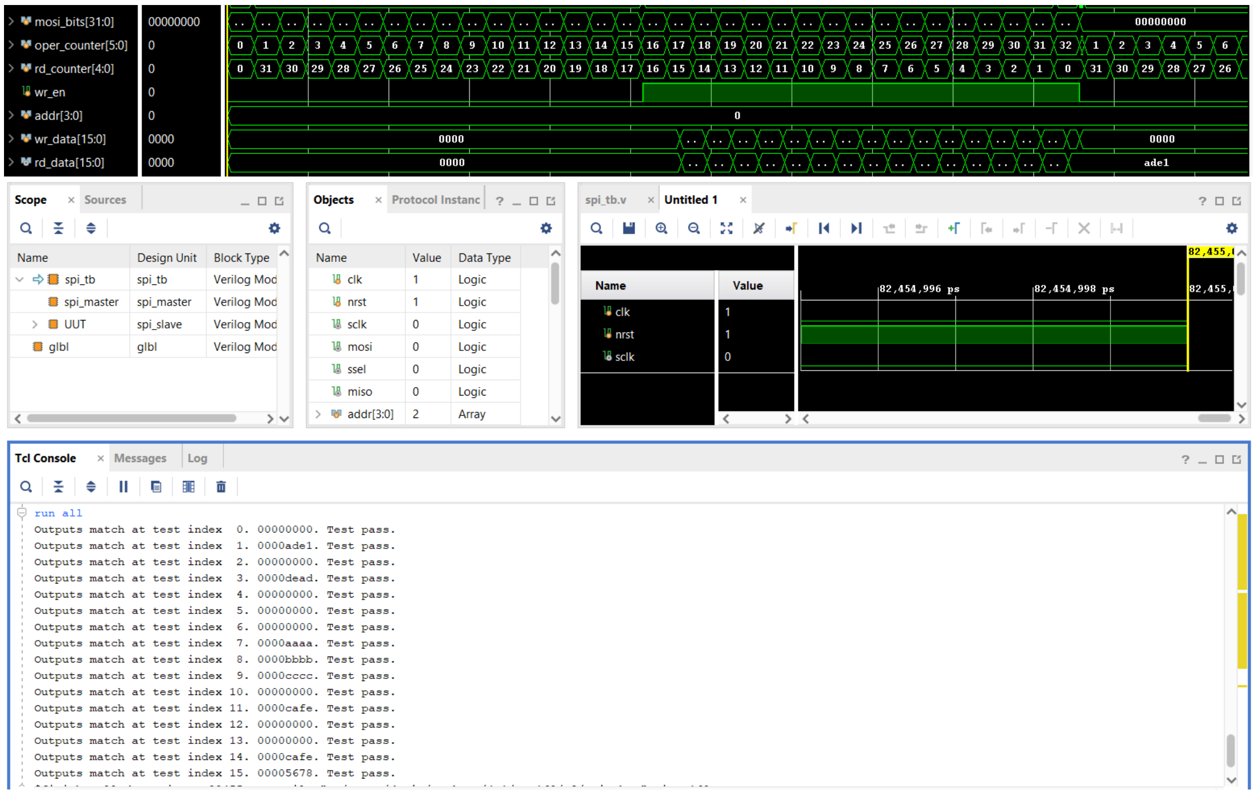
Name	Value
mosi_bits[31:0]	00000000
oper_counter[5:0]	00
rd_counter[4:0]	00
wr_en	0
addr[3:0]	0
wr_data[15:0]	0000
rd_data[15:0]	0000

0.000000 us
20.000000 us
40.000000 us
60.000000 us
80.000000 us
0, 8, a, b, c, a, b, c, 1, 2, 1, 2
ade1, 0000, dead, 0000, aaaa, bbbb, cccc, 0000, cafe, beef, 1234, 5678, 0000
0000..., 00000000, 00000000
2,1,2,2,1,1,c,b,a,c,b,a,8,8,0,0
0,0,1,1,0,1,0,0,0,1,1,1,0,1,0,1
0000,0000,5678,1234,beef,cafe,0000,0000,0000,cccc,bbbb,aaaa,0000,dead,0000,ade1
00005678,0000cafe,00000000,00000000,0000cafe,00000000,0000cccc,0000bbbb,0000aaaa,000000...
0 1 2 3 4 5 6 7 8 9 a b c d e f
ade1



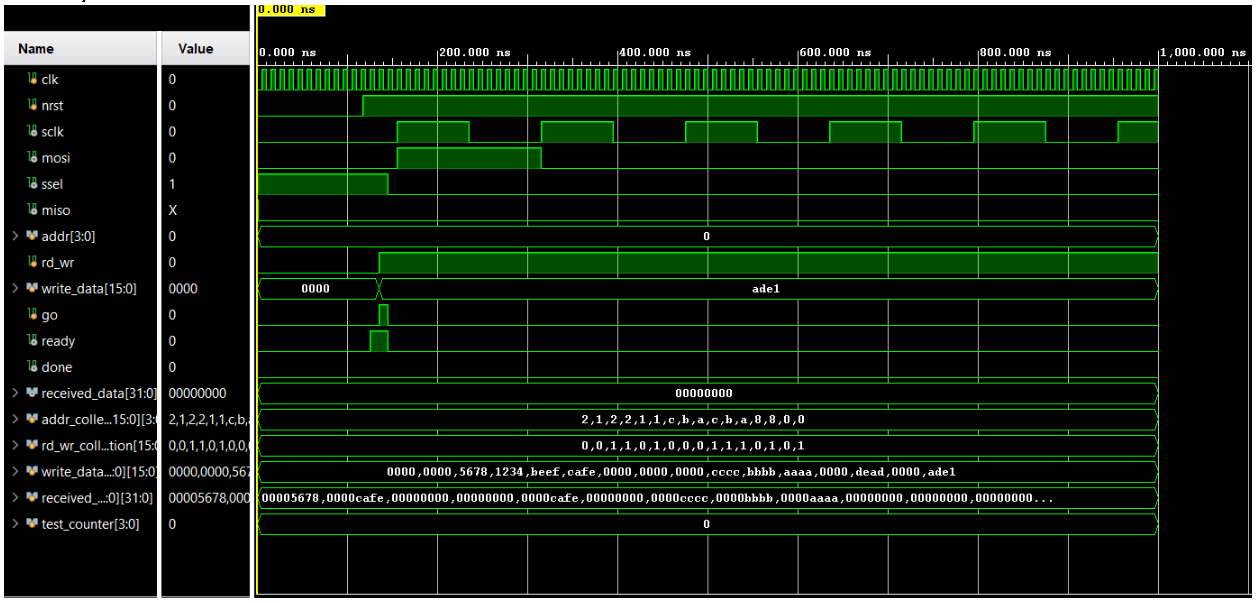
Name	Value
clk	0
nrst	0
sclk	0
mosi	0
ssel	1
miso	0
addr[3:0]	0
rd_wr	0
write_data[15:0]	0000
go	0
ready	0
done	0
received_data[31:0]	00000000
addr_colle...15:0][3:	2,1,2,2,1,1,c,b,
rd_wr_coll...tion[15:	0,0,1,1,0,1,0,0,
write_data...0][15:0]	0000,0000,567
received_...:0][31:0]	00005678,000
test_counter[3:0]	0
state[1:0]	0
mosi_bits[31:0]	00000000

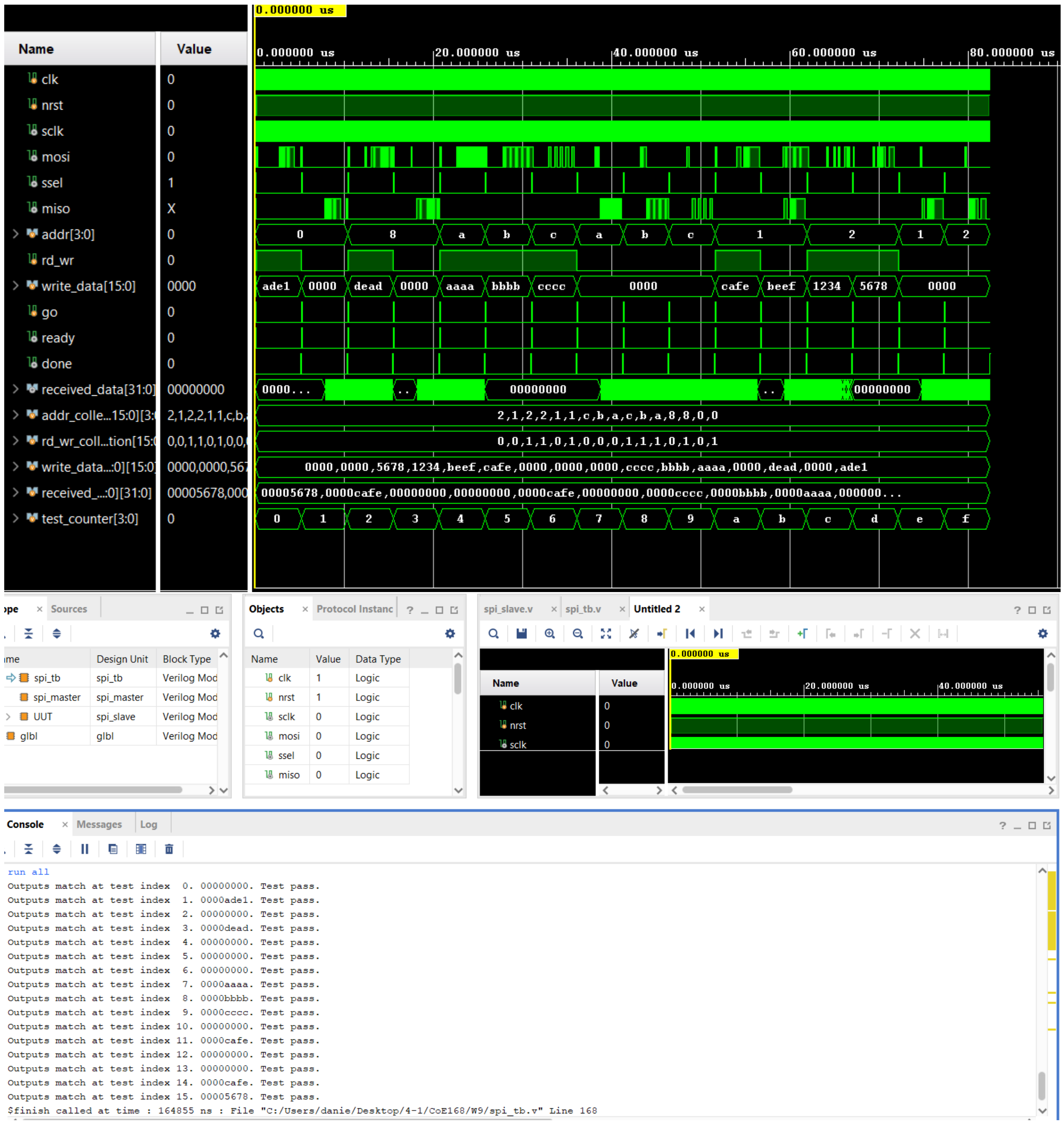
0.000 ns
1,000.000 ns
2,000.000 ns
3,000.000 ns
4,000.000 ns
5,000.000 ns
6,000.00
ade1
0000
00000000
2,1,2,2,1,1,c,b,a,c,b,a,8,8,0,0
0,0,1,1,0,1,0,0,0,1,1,1,0,1,0,1
0000,0000,5678,1234,beef,cafe,0000,0000,0000,cccc,bbbb,aaaa,0000,dead,0000,ade1
00005678,0000cafe,00000000,00000000,0000cafe,00000000,0000cccc,0000bbbb,0000aaaa,00000000,00000000,00000000,0000dead,00000000,0000ade1,0...
0 1 2 3 1



```
run all
Outputs match at test index  0. 00000000. Test pass.
Outputs match at test index  1. 0000ade1. Test pass.
Outputs match at test index  2. 00000000. Test pass.
Outputs match at test index  3. 0000dead. Test pass.
Outputs match at test index  4. 00000000. Test pass.
Outputs match at test index  5. 00000000. Test pass.
Outputs match at test index  6. 00000000. Test pass.
Outputs match at test index  7. 0000aaaa. Test pass.
Outputs match at test index  8. 0000bbbb. Test pass.
Outputs match at test index  9. 0000cccc. Test pass.
Outputs match at test index 10. 00000000. Test pass.
Outputs match at test index 11. 0000cafe. Test pass.
Outputs match at test index 12. 00000000. Test pass.
Outputs match at test index 13. 00000000. Test pass.
Outputs match at test index 14. 0000cafe. Test pass.
Outputs match at test index 15. 00005678. Test pass.
```

Post-Synthesis Functional Simulation Waveforms and Console





```
run all
Outputs match at test index  0. 00000000. Test pass.
Outputs match at test index  1. 0000ade1. Test pass.
Outputs match at test index  2. 00000000. Test pass.
Outputs match at test index  3. 0000dead. Test pass.
Outputs match at test index  4. 00000000. Test pass.
Outputs match at test index  5. 00000000. Test pass.
Outputs match at test index  6. 00000000. Test pass.
Outputs match at test index  7. 0000aaaa. Test pass.
Outputs match at test index  8. 0000bbbb. Test pass.
Outputs match at test index  9. 0000cccc. Test pass.
Outputs match at test index 10. 00000000. Test pass.
Outputs match at test index 11. 0000cafe. Test pass.
Outputs match at test index 12. 00000000. Test pass.
Outputs match at test index 13. 00000000. Test pass.
Outputs match at test index 14. 0000cafe. Test pass.
Outputs match at test index 15. 00005678. Test pass.
$finish called at time : 164855 ns : File "C:/Users/danie/Desktop/4-1/CoE168/W9/spi_tb.v" Line 168
```

spi_slave.v code

```
`timescale 1ns / 1ps

module spi_slave
(
    input clk,
    input nrst,
    input sclk,
    input mosi,
    input ssel,
    output reg miso
    );

reg [15:0] wr_data;
wire [15:0] rd_data;
reg [3:0] addr;
reg wr_en;

// write your other internal signals here
reg [1:0] state;
reg sclk_prev;
reg ssel_prev;
reg [31:0] mosi_bits;
reg [5:0] oper_counter;
reg [4:0] rd_counter;
reg wr_checker;

// register file instantiation
regfile regfile(
    .clk(clk),
    .nrst(nrst),
    .wr_en(wr_en),
    .addr(addr),
    .wr_data(wr_data),
    .rd_data(rd_data)
);

// you should only use posedge clk. Don't use posedge sclk, or negedge sclk!
always@(posedge clk or negedge nrst) begin
    if (!nrst) begin
        miso <= 0;
        wr_data <= 16'b0000000000000000;
        addr <= 4'b0000;
        wr_en <= 0;

        state <= 2'b00;
        sclk_prev <= 0;
        ssel_prev <= 0;
        mosi_bits <= 32'h00000000;
        oper_counter <= 6'b000000;
        rd_counter <= 6'b100000;
        wr_checker <= 0;

    end else begin
        sclk_prev <= sclk;  //01- posedge, won't update right away
        if (sclk_prev == 0 & sclk == 1) begin   //01
            mosi_bits <= {mosi_bits[30:0], mosi};
            oper_counter <= oper_counter + 1;
            rd_counter <= rd_counter - 1;
        end

        if (state == 2'b00) begin
            ssel_prev <= ssel;
            if (ssel_prev == 1 & ssel == 0) begin   //Ssel becomes 0 from 1
                state <= 2'b01;
                mosi_bits <= 32'h00000000;
                oper_counter <= 6'b000000;
                rd_counter <= 6'b100000;
            end
            else
                state <= 2'b00;
        end

        else if (state == 2'b01) begin
            if (oper_counter < 6'b010000)
                state <= 2'b01;
            else if (oper_counter == 6'b010000) begin   //Counter = 16, 16th bit
                state <= 2'b10;
                wr_en <= mosi_bits[15];
                addr <= mosi_bits[3:0];
                if (sclk_prev == 0 & sclk == 1 & mosi_bits[15] == 1)
                    wr_data <= {wr_data[14:0], mosi};
                else if (sclk_prev == 0 & sclk == 1 & mosi_bits[15] == 0)
                    miso <= rd_data[rd_counter];

            end
        end

        else if (state == 2'b10) begin  //Obtain 17th bit
            if (wr_en == 1) begin   //Write
                if (oper_counter < 6'b100000 & oper_counter > 6'b010000) begin
                    if (sclk_prev == 1 & sclk == 0)
                        wr_data <= {wr_data[14:0], mosi};
                    state <= 2'b10;
                end
                else if (oper_counter == 6'b100000 & oper_counter > 6'b010000) begin //Counter = 32, 32nd bit,
bit31
                    if (sclk_prev == 0 & sclk == 1)
                        wr_data <= {wr_data[14:0], mosi};
                    ssel_prev <= ssel;
                    state <= 2'b11;
                end
            end
            else if (wr_en == 0) begin  //Read
                if (oper_counter < 6'b100000 & oper_counter > 6'b010000) begin
                    //if (sclk_prev == 1 & sclk == 0)
```

```
                  miso <= rd_data[rd_counter];
               state <= 2'b10;
            end
            else if (oper_counter == 6'b100000 & oper_counter > 6'b010000) begin
               //if (sclk_prev == 0 & sclk == 1)
                  miso <= rd_data[rd_counter];
               ssel_prev <= ssel;
               state <= 2'b11;
            end
         end
      end

      else if (state == 2'b11) begin
         ssel_prev <= ssel;
         if (wr_en == 1) begin
            if (sclk_prev == 1 & sclk == 0)
               wr_data <= {wr_data[14:0], mosi};
         end
         else if (wr_en == 0) begin
            if (sclk_prev == 1 & sclk == 0)
               miso <= rd_data[rd_counter];
         end

         if (ssel_prev == 0 & ssel == 1) begin //Ssel becomes 1 from 0
            state <= 2'b00;
            miso <= 0;
            wr_data <= 16'b0000000000000000;
            addr <= 4'b0000;

            wr_en <= 0;
            state <= 2'b00;
            sclk_prev <= 0;
            mosi_bits <= 32'h00000000;
            oper_counter <= 6'b000000;
            rd_counter <= 6'b100000;
         end
         else
            state <= 2'b11;
      end

   end
end

endmodule
```