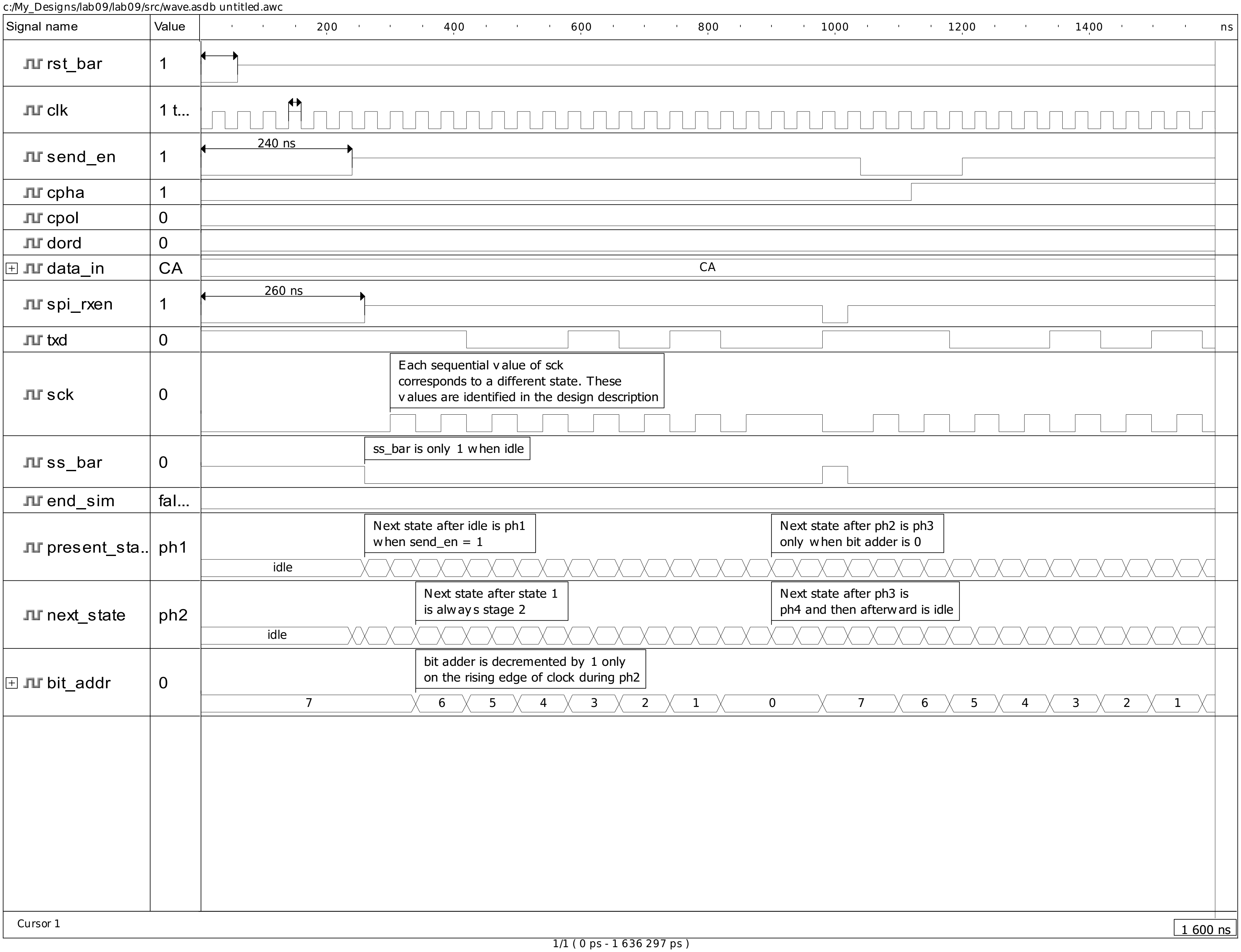
Lab 9: SPI Transmitter and Receiver

Ishabul Haque 111598085
Ken Ejinkonye 112011486

Section 2
05/04/20

```vhdl
-------------------------------------------------------------------------------
--
-- Title       : spi_tx_shifter_tb
-- Design      : spi_tx_shifter_tb
-- Author      : Ishabul Haque and Ken Ejinkonye
-- Company     : Stony Brook
--
-------------------------------------------------------------------------------
--
-- File        : c:\My_Designs\Lab_09\SPI_Transmitter_and_Reciever\src\spi_tx_shifter_td
-- Generated   : Sun May  3 10:51:17 2020
-- From        : interface description file
-- By          : Itf2Vhdl ver. 1.22
--
-------------------------------------------------------------------------------
--
-- Description : Non-self checking testbench for spi_tx_shifter design
--
-------------------------------------------------------------------------------


library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
library work;
use work.all;


entity spi_tx_shifter_tb is
end spi_tx_shifter_tb;



architecture tx_shifter_tb of spi_tx_shifter_tb is
   --stimulus signals
    signal rst_bar : std_logic;
    signal clk : std_logic;
    signal send_en: std_logic;
    signal cpha: std_logic;
    signal cpol: std_logic;
    signal dord: std_logic;
    signal data_in: std_logic_vector(7 downto 0);
    signal spi_rxen: std_logic;

    --observed signals
    signal txd: std_logic;
    signal sck: std_logic;
    signal ss_bar: std_logic;


    constant period : time := 40ns;
    signal end_sim : boolean := false;
```

```

begin
    -- Unit Under Test port map
    UUT: entity spi_tx_shifter
    port map (
        rst_bar => rst_bar,
        clk => clk,
        send_en => send_en,
        cpha => cpha,
        cpol => cpol,
        dord => dord,
        data_in => data_in,
        txd => txd,
        sck => sck,
        ss_bar => ss_bar,
        spi_rxen => spi_rxen
        );

     -- Process to generate value for input data_in
     data_gen:process
     begin
         for i in 1 to 2 loop
             wait until clk='0';
        data_in<="11001010";
        end loop;
        wait;
     end process;

     -- Process to generate clock

     clock_gen:process
     begin
         clk<='0';
         loop
             wait for period/2;
             clk<=not clk;
             exit when end_sim = true;
        end loop;
        wait;
     end process;

    -- Process to generate values for reset bar
    reset: process
    begin
        -- Reset bar intially 0 for 60ns
        rst_bar <='0';
        for i in 1 to 2 loop
            wait until clk = '1';
        end loop;
        rst_bar<='1';
        wait;
    end process;

    -- Process to generate values for dord,cpol,cpha. The push button
    -- is represented by design send_pos_edge and cpol and cpha are
    -- not changed in the middle of a transmission. Value of dord
    -- determines which bit position the data is shifted to.
    push_button:process
    begin
        -- Initialized values
        send_en <= '0';
        cpol <= '0';
        cpha <= '0';
        dord <= '0';

        wait for 4*period;
        for i in 0 to 7 loop
            -- Generates the different modes of operation
            (dord, cpol, cpha) <= to_unsigned(i,3);
            wait for 2*period;
            send_en <= '1';
            wait for 20*period;
            send_en <= '0';
            wait for 2*period;

        end loop;
        end_sim <= true;
        wait;
    end process;


end tx_shifter_tb;

```



c:/My_Designs/lab09/lab09/src/wave.asdb untitled.awc

Signal name	Value
rst_bar	1
clk	1 t...
send_en	1
cpha	1
cpol	0
dord	0
data_in	CA
spi_rxen	1
txd	0
sck	0
ss_bar	0
end_sim	fal...
present_sta..	ph1
next_state	ph2
bit_addr	0

Each sequential v alue of sck corresponds to a different state. These v alues are identified in the design description

ss_bar is only 1 w hen idle

Next state after idle is ph1 w hen send_en = 1

Next state after ph2 is ph3 only w hen bit adder is 0

Next state after state 1 is alw ay s stage 2

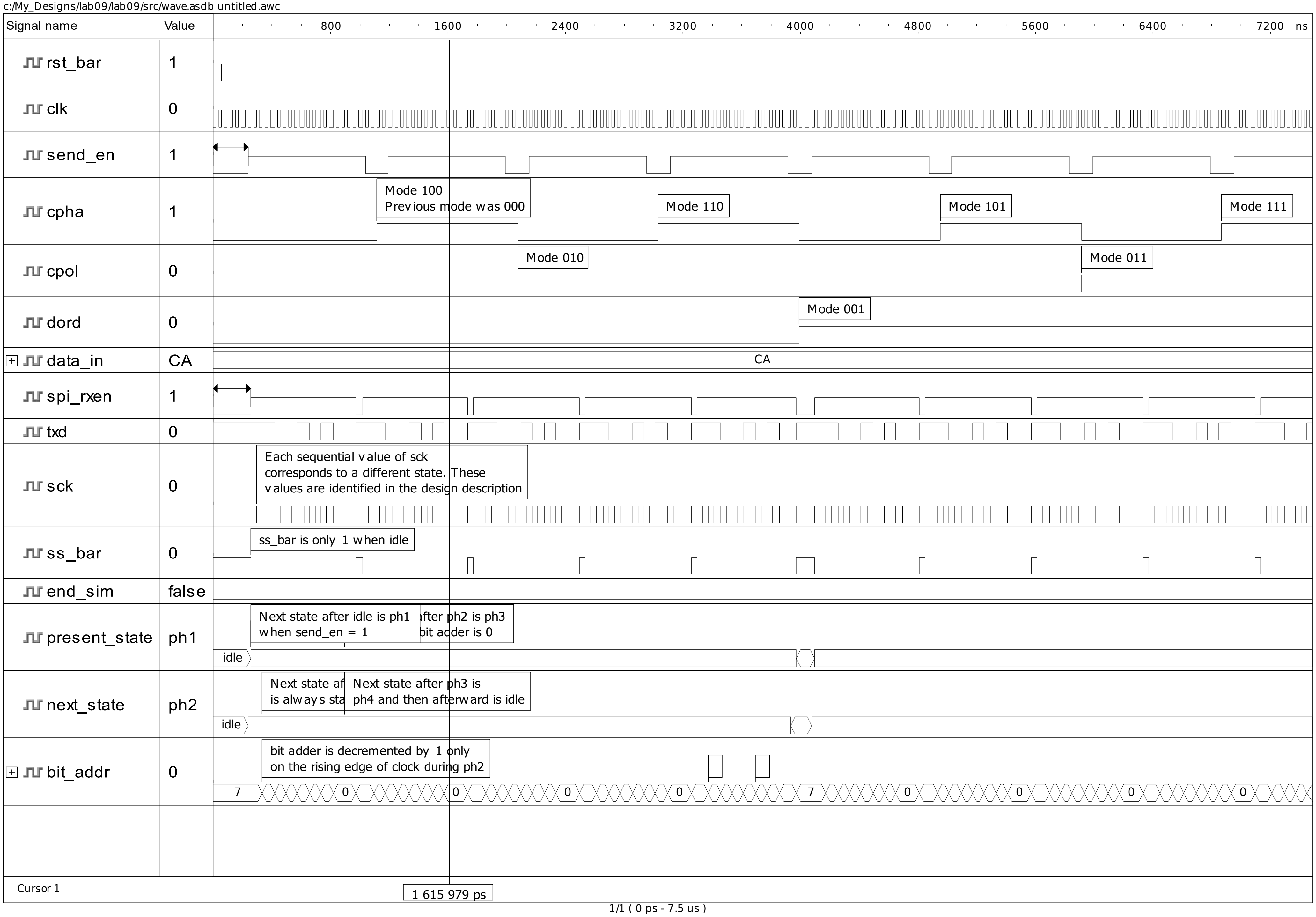
Next state after ph3 is ph4 and then afterw ard is idle

bit adder is decremented by 1 only on the rising edge of clock during ph2

Cursor 1

1 600 ns

1/1 (0 ps - 1 636 297 ps)



c:/My_Designs/lab09/lab09/src/wave.asdb untitled.awc

Signal name	Value
rst_bar	1
clk	0
send_en	1
cpha	1
cpol	0
dord	0
data_in	CA
spi_rxen	1
txd	0
sck	0
ss_bar	0
end_sim	false
present_state	ph1
next_state	ph2
bit_addr	0

Mode 100
Previous mode was 000

Mode 110

Mode 101

Mode 111

Mode 010

Mode 011

Mode 001

Each sequential value of sck corresponds to a different state. These values are identified in the design description

ss_bar is only 1 when idle

Next state after idle is ph1 when send_en = 1

fter ph2 is ph3
bit adder is 0

Next state af
is always sta

Next state after ph3 is ph4 and then afterward is idle

bit adder is decremented by 1 only on the rising edge of clock during ph2

Cursor 1

1 615 979 ps

1/1 (0 ps - 7.5 us)

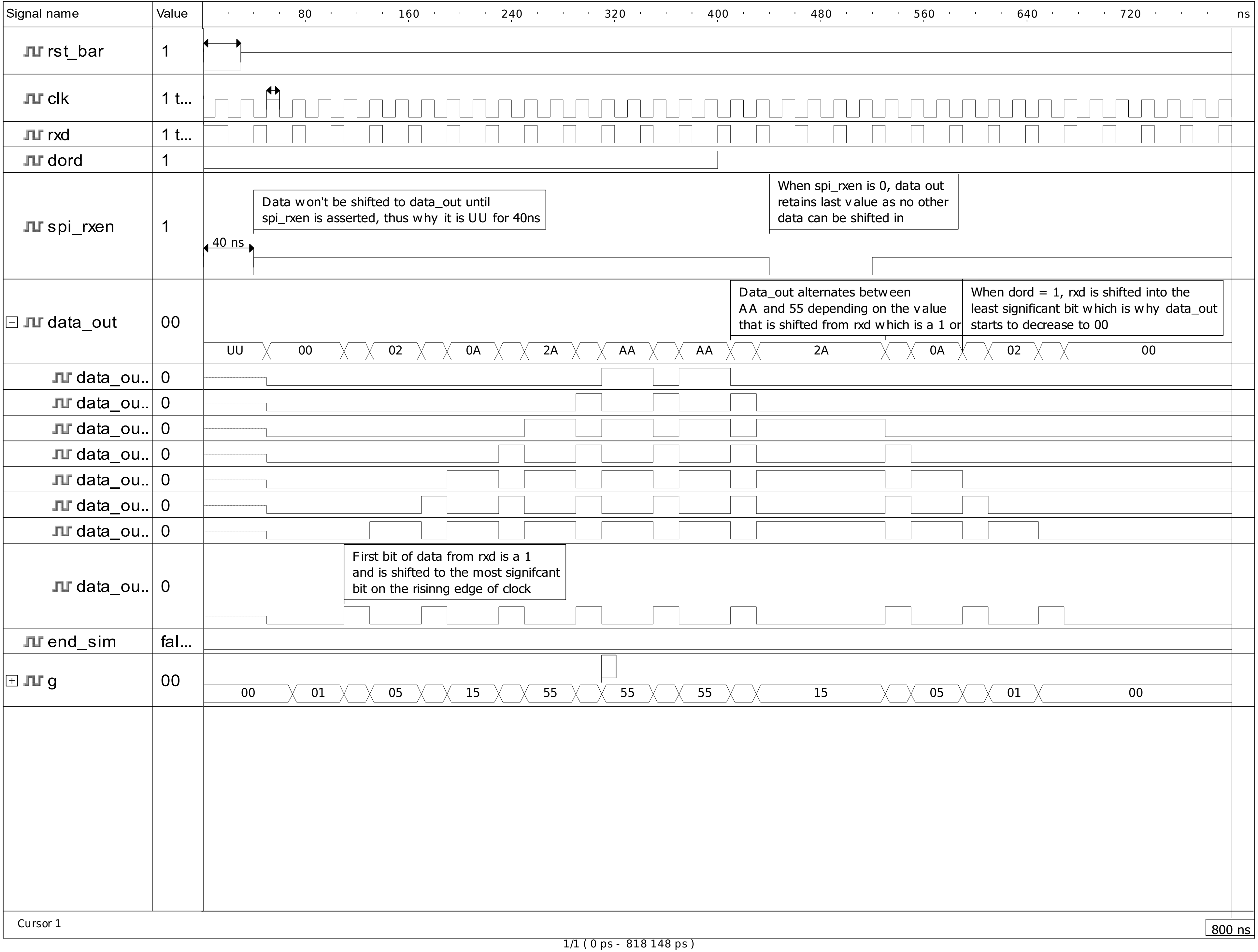
File: C:/My_Designs/lab09/lab09/src/spi_rx_shifter.vhd

```
--------------------------------------------------------------------------------
--
-- Title       : spi_rx_shifter
-- Design      : task1_2
-- Author      : Ishabul Haque and Ken Ejinkonye
-- Company     : Stony Brook
--
-- Description : The receiver shifter spi_rx_shifter converts the serial data
-- at its rxd input to parallel and and provides this parallel result as output
-- data_out. This data may arrive most significant bit first or least significant
-- bit first, as determined by the dord input.The final parallel value at data_out
-- must always have its most significant bit in the leftmost position.

-- List of Circuit Features To Be Verified:
-- Inputs: rxd, rst_bar, clk,spi_rxen,dord

-- Outputs: data_out

--------------------------------------------------------------------------------

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


entity spi_rx_shifter is
    port(
    rxd : in std_logic;         -- data received from slave
    rst_bar : in std_logic;     -- asynchronous reset
    clk : in std_logic;         -- system clock
    spi_rxen : in std_logic;    -- signal to enable shift
    dord : in std_logic;        -- data order bit
    data_out : out std_logic_vector(7 downto 0) -- received data
    );
end spi_rx_shifter;


--}} End of automatically maintained section

architecture rx_shifter of spi_rx_shifter is
-- Signal g is a temporary vector to manipulate value going to data_out
signal g : std_logic_vector(7 downto 0):= "00000000";




begin

    dord_change: process(dord, rxd, spi_rxen,g)
    begin
        -- Data can only be shifted when spi_rxen is asserted
        if spi_rxen = '1' then
            -- If dord = 0, data is shifted to most significant bit
            if (dord='0') then
                -- Data is only shifted at the rising edge of clock
                if rising_edge(clk) then
                    -- Data being shifted in is the value of rxd
                    -- Initial bit position will always equal rxd
                    data_out(0) <= rxd;
                    g(0) <= rxd;
                    -- For loop to implement shifting method using g
                    -- as a test vector to manipulate data
                    for i in 7 downto 0 loop
                        if i>0 then
                            g(i)<=g(i-1);
                            data_out<=g;
                        end if;


                    end loop;
                end if ;
            -- If dord = 1, data is shifted to least significant bit
            elsif (dord='1') then
                if rising_edge(clk) then

                    data_out(7)<=rxd;
                    g(0) <= rxd;
                    data_out <= g;

                    for i in 7 downto 0 loop

                        if i>0 then
                            g(i-1)<=g(i);
                            data_out<=g;
                        end if;

                    end loop;
                end if;


            end if;
        end if;

    end process;

end rx_shifter;

```

```vhdl
-------------------------------------------------------------------------------
--
-- Title       : spi_rx_shifter_tb
-- Design      : spi_rx_shifter_tb
-- Author      : Ishabul Haque and Ken Ejinkonye
-- Company     : Stony Brook University
--
-------------------------------------------------------------------------------
--
-- File        : c:\My_Designs\Lab_09\SPI_Transmitter_and_Reciever\src\spi_rx_shifter_td
-- Generated   : Tue May  5 10:55:14 2020
-- From        : interface description file
-- By          : Itf2Vhdl ver. 1.22
--
-------------------------------------------------------------------------------
--
-- Description : Non self checking test bench for spi_rx_shifter design
--
-------------------------------------------------------------------------------


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



entity spi_rx_shifter_tb is
end spi_rx_shifter_tb;



architecture rx_shifter_tb of spi_rx_shifter_tb is

    --stimulus signals
    signal rst_bar : std_logic;
    signal clk : std_logic;
    signal rxd : std_logic;
    signal dord: std_logic;
    signal spi_rxen: std_logic;

    --observed signals
    signal data_out: std_logic_vector(7 downto 0);

    -- Constants
    constant period : time := 20ns;
    signal end_sim : boolean := false;
begin
    -- Unit Under Test port map
    UUT: entity spi_rx_shifter
    port map (
        rst_bar => rst_bar,
        clk => clk,
        rxd => rxd,
        dord => dord,
        data_out => data_out,
        spi_rxen => spi_rxen
        );

        -- Process to generate value for clock
        clock_gen:process

        begin

            clk<='0';
            loop
                wait for period/2;
                clk<=not clk;
                exit when end_sim = true;
            end loop;
            wait;
        end process;

        -- Process to generate value for reset bar
        reset: process
        begin

            rst_bar<='0';
            for i in 1 to 2 loop
                wait until clk = '1';

            end loop;
            rst_bar<='1';
            wait;

        end process;

        -- Process to generate dord, dord simply changes
        -- from 0 to 1 for verification purposes of the design
        dord_gen: process
        begin
            loop
            dord <= '0';
            wait for 400 ns;
            dord <= '1';
            exit when end_sim = true;
            end loop;
        end process;

        -- Process to generate value for spi_rxen
        -- A value of 0 is asserted to verify data
        -- is only shifted when spi_rxen is asserted
        spi_rxen_gen: process

        begin

            spi_rxen <= '0';

            loop
```

```
                wait for 2*period;
                spi_rxen <= '1';
                wait for 20*period;
                spi_rxen <= '0';
                wait for 2*period;
                exit when end_sim = true;
            end loop;

        end process;

        -- Proocess to generate value for rxd, data
        -- being shifted in
        rxd_gen : process
        begin
            rxd <= '1';
            loop
                wait for period/2;
                rxd <= '1';
                wait for period/2 ;
                rxd <= '0';
                wait for period/2;
                exit when end_sim = true;
            end loop;
        end process;



end rx_shifter_tb;
```

c:/My_Designs/lab09/lab09/src/wave.asdb untitled.awc



Data won't be shifted to data_out until spi_rxen is asserted, thus why it is UU for 40ns

When spi_rxen is 0, data out retains last value as no other data can be shifted in

Data_out alternates between AA and 55 depending on the value that is shifted from rxd which is a 1 or

When dord = 1, rxd is shifted into the least significant bit which is why data_out starts to decrease to 00

First bit of data from rxd is a 1 and is shifted to the most signifcant bit on the risinng edge of clock

1/1 (0 ps - 818 148 ps)

File: C:/My_Designs/lab09/lab09/src/send_pos_edge_det.vhd

```vhdl
--------------------------------------------------------------------------------
--
-- Title       : send_pos_edge_det
-- Design      : send_pos_edge_det
-- Author      : Ishabul Haque and Ken Ejinkonye
-- Company     : Stony Brook
--
-- Description : A positive edge detector is used to detect the
-- positive edge of the send and to generate a narrow pulse that will
-- be used by the spi_tx_shifter to determine when it should start to send
-- a byte.
-- List of Circuit Features To Be Verified:
-- Inputs: rst_bar, clk, send
-- Outputs: send_en


--------------------------------------------------------------------------------

library IEEE;
use IEEE.std_logic_1164.all;

entity send_pos_edge_det is
    port(
    rst_bar : in std_logic;     -- asynnchronous system reset
    clk : in std_logic;         -- system clock
    send : in std_logic;        -- debounced send input
    send_en : out std_logic     -- send enable output pulse
    );
end send_pos_edge_det;


architecture moore_fsm of send_pos_edge_det is
    type state is (state_a, state_b, state_c);
    signal present_state, next_state : state;

    begin
        state_reg: process (clk,rst_bar)
        begin
            if rst_bar = '0' then
            -- If rst_bar is enabled, then present state will switch
            -- to state a
                present_state <= state_a;
            elsif rising_edge(clk) then
            -- If rst_bar is not enabled, the present state will switch
            -- to the next state on the rising edge of clock
                present_state <= next_state;
            end if;
        end process;

        outputs: process (present_state)
        begin
            case present_state is
                -- Output send_en will only be high when the present state
                -- is in state c, otherwise it will output low
                when state_c => send_en <= '1';
                when others => send_en <= '0';
            end case;
        end process;

        nxt_state: process (present_state, send)
        begin
            -- Moore FSM, state values are determined by
            -- state diagram
            case present_state is
                when state_a =>
                if send = '0' then
                    next_state <= state_b;
                else
                    next_state <= state_b;
                end if;

                when state_b =>
                if send = '1' then
                    next_state <= state_c;
                else
                    next_state <= state_b;
                end if;

                when others =>
                if send = '0' then
                    next_state <= state_b;
                else
                    next_state <= state_a;
                end if;
            end case;
        end process;
    end moore_fsm;



```

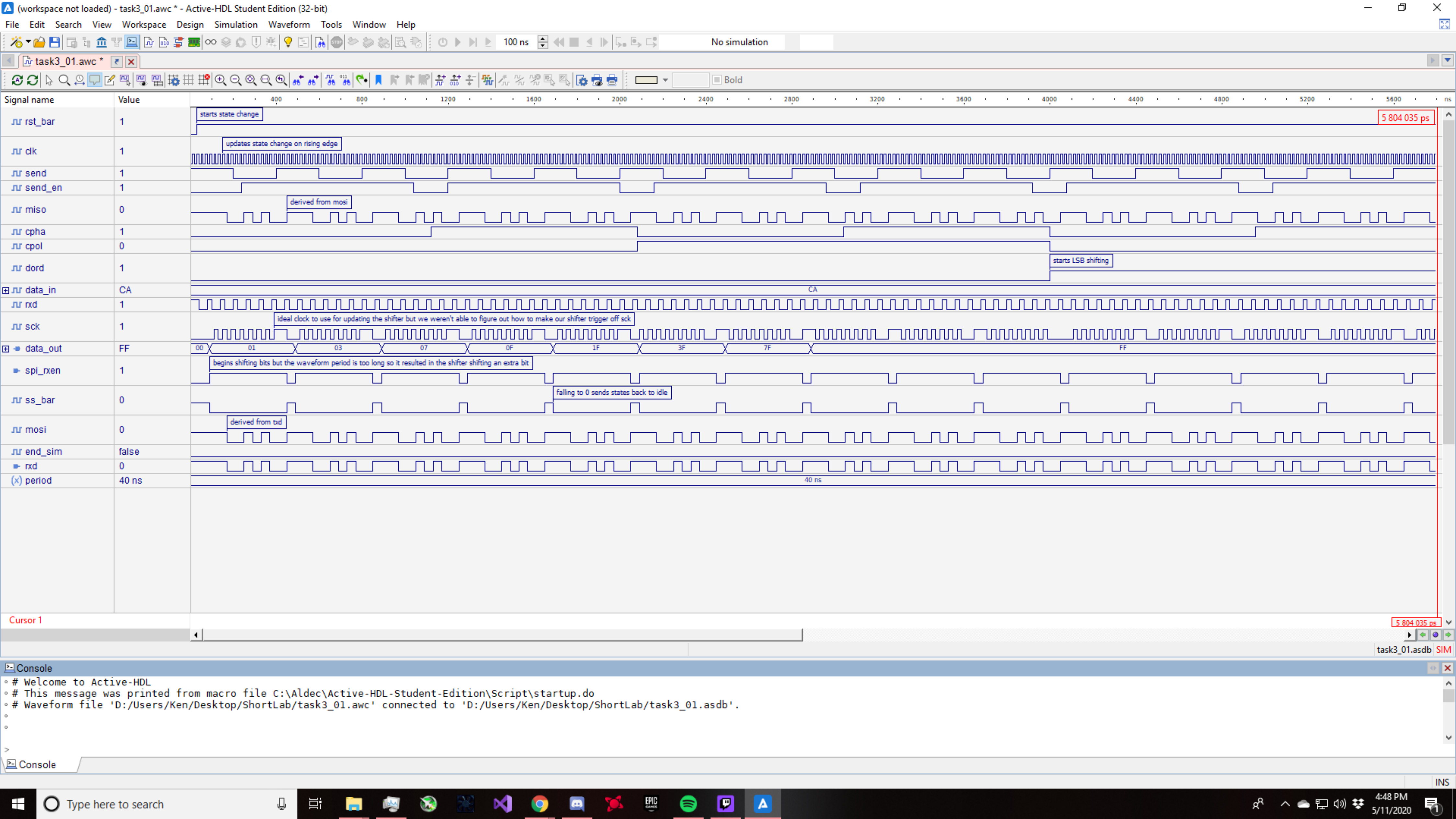
```
--------------------------------------------------------------------------------
--
-- Title       : SPI_test_system_II
-- Design      : SPI_Transmitter_and_Reciever
-- Author      : kenechukwu.ejinkonye@stonybrook.edu
-- Company     : Stony Brook University
--
--------------------------------------------------------------------------------
--
-- File        : c:\My_Designs\Lab_09\SPI_Transmitter_and_Reciever\src\spi_tx_shifter_tb.vhd
-- Generated   : Sun May  3 10:51:17 2020
-- From        : interface description file
-- By          : Itf2Vhdl ver. 1.22
--
--------------------------------------------------------------------------------
--
-- Description :  SPI Test System II Top Level
--
--------------------------------------------------------------------------------

--{{ Section below this comment is automatically maintained
--   and may be overwritten
--{entity {spi_tx_shifter_tb} architecture {spi_tx_shifter_tb}}

library ieee;
use ieee.numeric_std.all;
use ieee.std_logic_1164.all;
library work;
use work.all;

entity SPI_test_system_II is
    port(
    rst_bar : in std_logic;     -- asynchronous system reset
    clk : in std_logic;         -- system clock
    send : in std_logic;        -- positive pulse to start transmission
    cpol : in std_logic;        -- clock polarity setting
    cpha : in std_logic;        -- clock phase setting
    dord : in std_logic;        -- transmission data order 0 => msb first
    miso : in std_logic;        -- master in slave out
    spi_rxen : in std_logic;    -- signal to enable shift
    data_in : in std_logic_vector(7 downto 0);      -- parallel input data
    data_out : out std_logic_vector(7 downto 0);    -- parallel output data
    mosi : out std_logic;       -- master out slave in SPI serial data
    sck : out std_logic;        -- SPI shift clock to slave
    ss_bar : out std_logic      -- slave select signal );
    );




end SPI_test_system_II;
```

```

architecture structural of SPI_test_system_II is

signal temp_send_en : std_logic;
signal temp_spi_rxen : std_logic;





begin


-- Port Map for first two designs to structural design SPI_test_system
u0 : entity send_pos_edge_det port map(rst_bar => rst_bar, clk => clk,
    send => send, send_en => temp_send_en);

u1: entity spi_tx_shifter port map(rst_bar => rst_bar, clk => clk,
    cpol => cpol, cpha => cpha,send_en => temp_send_en, dord => dord,
    data_in => data_in,txd => mosi,sck => sck, ss_bar => ss_bar,
    spi_rxen  => temp_spi_rxen );


u2: entity spi_rx_shifter port map(rst_bar => rst_bar, clk => clk,
    rxd => miso, dord => dord, data_out => data_out, spi_rxen  => temp_spi_rxen);




end structural;
```

```
--------------------------------------------------------------------------------
--
-- Title       : SPI_test_system_II_tb
-- Design      : SPI_test_system_II_tb
-- Author      : Ishabul Haque and Ken Ejinkonye
-- Company     : Stony Brook
--
--------------------------------------------------------------------------------
--
-- File        : c:\My_Designs\Lab_09\SPI_Transmitter_and_Reciever\src\spi_tx_shifter_td
-- Generated   : Sun May  3 10:51:17 2020
-- From        : interface description file
-- By          : Itf2Vhdl ver. 1.22
--
--------------------------------------------------------------------------------
--
-- Description : Non-self checking testbench for spi_tx_shifter design
--
--------------------------------------------------------------------------------


library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
library work;
use work.all;


entity SPI_test_system_II_tb is
end SPI_test_system_II_tb;



architecture SPI_test_system_II_tb of SPI_test_system_II_tb is
   --stimulus signals
    signal rst_bar : std_logic;
    signal clk : std_logic;
    signal send : std_logic;        -- positive pulse to start transmission
    signal send_en: std_logic;
    signal miso : std_logic;        -- master in slave out
    signal cpha: std_logic;
    signal cpol: std_logic;
    signal dord: std_logic;
    signal data_in: std_logic_vector(7 downto 0);
    signal spi_rxen: std_logic;
    signal rxd : std_logic;

    --observed signals
    signal txd: std_logic;
    signal sck: std_logic;
    signal ss_bar: std_logic;
    signal data_out :  std_logic_vector(7 downto 0);    -- parallel output data
    signal mosi :  std_logic;       -- master out slave in SPI serial data


    constant period : time := 40ns;
    signal end_sim : boolean := false;
    --constant data_in : std_logic_vector(7 downto 0) := x;
    signal loopback : std_logic;


begin
    -- Unit Under Test port map
    UUT: entity SPI_test_system_II
    port map (
        rst_bar => rst_bar,
        clk => clk,
        --send_en => send_en,
        send => send,
        miso => miso,
        cpha => cpha,
        spi_rxen => spi_rxen,   -- signal to enable shift
        cpol => cpol,
        dord => dord,
        data_in => data_in,
        mosi => mosi,
        sck => sck,
        ss_bar => ss_bar
        --spi_rxen => spi_rxen,
        );

        loopback <= mosi;
        miso <= loopback;

        -- Duration of signal send is 200ns at a high value
        send_gen: process
        begin
            send <= '1';
            for i in 0 to 28 loop      --28 rising edges
                wait for 200ns;        -- Send pulse duration of 200ns
                send <= not send;
            end loop;                  --end pulse

            std.env.finish;
        end process;



     -- Process to generate value for input data_in
     data_gen:process
     begin
         for i in 1 to 2 loop
             wait until clk='0';
        data_in <= "11001010";
        end loop;
        wait;
     end process;
```

```vhdl

 -- Proocess to generate clock
 clock_gen:process
 begin
     clk<='0';
     loop
         wait for period/4;
         clk<=not clk;
         exit when end_sim = true;
    end loop;
    wait;
 end process;

-- Process to generate values for reset bar
reset: process
begin
    -- Reset bar intially 0 for 60ns
    rst_bar<='0';
    for i in 1 to 2 loop
        wait until clk = '1';
    end loop;
    rst_bar<='1';
    wait;
end process;

-- Proocess to generate value for rxd, data
    -- being shifted in
    rxd_gen : process
    begin
        rxd <= '1';
        loop
            wait for period/2;
            rxd <= '1';
            wait for period/2 ;
            rxd <= '0';
            wait for period/2;
            exit when end_sim = true;
        end loop;
    end process;

    dord_gen: process
    begin
        loop
        dord <= '0';
        wait for 400 ns;
        dord <= '1';
        exit when end_sim = true;
        end loop;
    end process;





-- Process to generate values for dord,cpol,cpha. The push button
-- is represented by design send_pos_edge and cpol and cpha are
-- not changed in the middle of a transmission. Value of dord
```

```vhdl
    -- determines which bit position the data is shifted to.
    push_button:process
    begin
        -- Initialized values
        send_en <= '0';
        cpol <= '0';
        cpha <= '0';
        dord <= '0';

        wait for 4*period;
        for i in 0 to 9 loop
            -- Generates the different modes of operation
            (dord, cpol, cpha) <= to_unsigned(i,3);
            wait for 2*period;
            send_en <= '1';
            wait for 20*period;
            send_en <= '0';
            wait for 2*period;

        end loop;
        dord <= '1';
        end_sim <= true;
        wait;
    end process;

end SPI_test_system_II_tb;

```


(workspace not loaded) - task3_01.awc * - Active-HDL Student Edition (32-bit)
File Edit Search View Workspace Design Simulation Waveform Tools Window Help
100 ns
No simulation
task3_01.awc *
Bold
400 800 1200 1600 2000 2400 2800 3200 3600 4000 4400 4800 5200 5600 ns
5 804 035 ps
CA
00 01 03 07 0F 1F 3F 7F FF
40 ns
Cursor 1
task3_01.asdb SIM
Console
INS
Type here to search
4:48 PM
5/11/2020

Signal name	Value
rst_bar	1
clk	1
send	1
send_en	1
miso	0
cpha	1
cpol	0
dord	1
data_in	CA
rxd	1
sck	1
data_out	FF
spi_rxen	1
ss_bar	0
mosi	0
end_sim	false
rxd	0
period	40 ns

starts state change

updates state change on rising edge

derived from mosi

starts LSB shifting

ideal clock to use for updating the shifter but we weren't able to figure out how to make our shifter trigger off sck

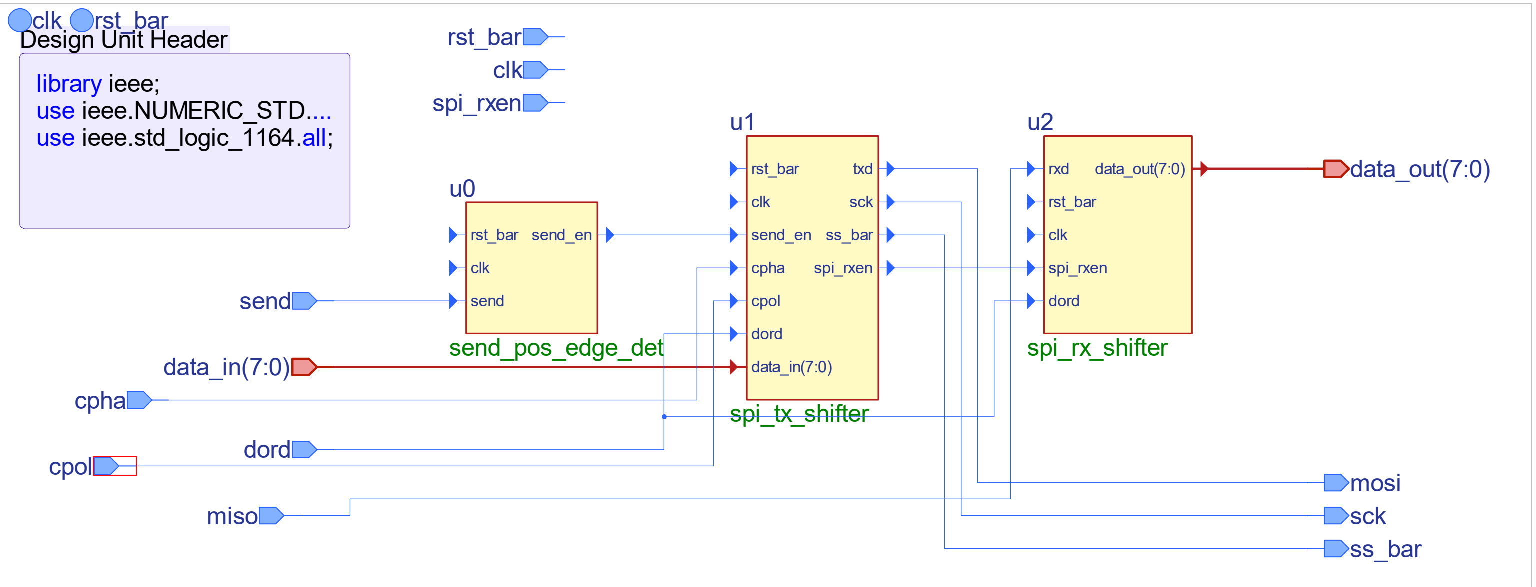
begins shifting bits but the waveform period is too long so it resulted in the shifter shifting an extra bit

falling to 0 sends states back to idle

derived from txd

```
# Welcome to Active-HDL
# This message was printed from macro file C:\Aldec\Active-HDL-Student-Edition\Script\startup.do
# Waveform file 'D:/Users/Ken/Desktop/ShortLab/task3_01.awc' connected to 'D:/Users/Ken/Desktop/ShortLab/task3_01.asdb'.
```

Project lab09



```
library ieee;
use ieee.NUMERIC_STD....
use ieee.std_logic_1164.all;
```

	(C)ALDEC. Inc **2260 Corporate Circle** **Henderson, NV 89074**
Created:	5/11/20
Title:	No Title
Revision:	**1.0**
Page:	1 / 1

```
      Lattice Mapping Report File for Design Module 'SPI_test_system_II'
```

Design Information

```
Command line:   map -a LatticeXP -p LFXP3C -t TQFP100 -s 4 -oc Commercial
     lab09_impl1.ngd -o lab09_impl1_map.ncd -pr lab09_impl1.prf -mp
     lab09_impl1.mrp -lpf
     C:/lscc/diamond/3.11_x64/bin/nt64/impl1/lab09_impl1_synplify.lpf -lpf
     C:/lscc/diamond/3.11_x64/bin/nt64/lab09.lpf -gui -msgset
     C:/lscc/diamond/3.11_x64/bin/nt64/promote.xml
Target Vendor:  LATTICE
Target Device:  LFXP3CTQFP100
Target Performance:   4
Mapper:  mg5g00,  version:  Diamond (64-bit) 3.11.2.446
Mapped on:  05/11/20  16:21:22
```

Design Summary

```
   Number of registers:     20 out of  3258 (1%)
      PFU registers:           20 out of  3072 (1%)
      PIO registers:            0 out of   186 (0%)
   Number of SLICEs:        17 out of  1536 (1%)
      SLICEs as Logic/ROM:     17 out of  1536 (1%)
      SLICEs as RAM:            0 out of   384 (0%)
      SLICEs as Carry:          0 out of  1536 (0%)
   Number of LUT4s:         25 out of  3072 (1%)
      Number used as logic LUTs:         25
      Number used as distributed RAM:     0
      Number used as ripple logic:        0
      Number used as shift registers:     0
   Number of PIO sites used: 25 out of 62 (40%)
   Number of PIO FIXEDDELAY:    0
   Number of DQSDLLs:  0 out of 2 (0%)
   Number of PLLs:  0 out of 2 (0%)
   Number of block RAMs:  0 out of 6 (0%)
   Number of GSRs:  1 out of 1 (100%)
   JTAG used :      No
   Readback used :  No
   Oscillator used :  No
   Startup used :   No
   Notes:-
      1. Total number of LUT4s = (Number of logic LUT4s) + 2*(Number of
     distributed RAMs) + 2*(Number of ripple logic)
      2. Number of logic LUT4s does not include count of distributed RAM and
     ripple logic.
   Number of clocks:  3
     Net u1.present_state[4]: 4 loads, 0 rising, 4 falling (Driver:
     u1/present_state[4] )
     Net clk_c: 8 loads, 8 rising, 0 falling (Driver: PIO clk )
     Net u1/N_52_lo: 2 loads, 2 rising, 0 falling (Driver:
     u1/present_state_RNIRO9D[0] )
   Number of Clock Enables:  0
   Number of local set/reset loads for net rst_bar_c merged into GSR:  15
   Number of LSRs:  1
     Net u1/un3_rst_bar: 2 loads, 2 LSLICEs
   Number of nets driven by tri-state buffers:  0
   Top 10 highest fanout non-clock nets:
     Net u1/bit_addr[2]: 9 loads

     Net dord_c: 8 loads
     Net u1/bit_addr[1]: 7 loads
     Net u1/bit_addr[0]: 6 loads
     Net u1/present_state[2]: 6 loads
     Net temp_send_en: 4 loads
     Net data_out_c[1]: 3 loads
     Net data_out_c[2]: 3 loads
     Net data_out_c[3]: 3 loads
     Net data_out_c[4]: 3 loads




   Number of warnings:  3
   Number of errors:    0
```

Design Errors/Warnings

```
WARNING - map: Using local reset signal 'rst_bar_c' to infer global GSR net.
WARNING - map: IO buffer missing for top level port cpha...logic will be
     discarded.
WARNING - map: IO buffer missing for top level port spi_rxen...logic will be
     discarded.
```

IO (PIO) Attributes

```
+---------------------+-----------+-----------+------------+------------+
| IO Name             | Direction | Levelmode | IO         | FIXEDDELAY |
|                     |           |  IO_TYPE  | Register   |            |
+---------------------+-----------+-----------+------------+------------+
| data_out[0]         | OUTPUT    | LVCMOS25  |            |            |
+---------------------+-----------+-----------+------------+------------+
| rst_bar             | INPUT     | LVCMOS25  |            |            |
+---------------------+-----------+-----------+------------+------------+
| ss_bar              | OUTPUT    | LVCMOS25  |            |            |
+---------------------+-----------+-----------+------------+------------+
| sck                 | OUTPUT    | LVCMOS25  |            |            |
+---------------------+-----------+-----------+------------+------------+
| mosi                | OUTPUT    | LVCMOS25  |            |            |
+---------------------+-----------+-----------+------------+------------+
| data_out[7]         | OUTPUT    | LVCMOS25  |            |            |
+---------------------+-----------+-----------+------------+------------+
| data_out[6]         | OUTPUT    | LVCMOS25  |            |            |
+---------------------+-----------+-----------+------------+------------+
| data_out[5]         | OUTPUT    | LVCMOS25  |            |            |
| data_out[5]         | OUTPUT    | LVCMOS25  |            |            |
+---------------------+-----------+-----------+------------+------------+
| data_out[4]         | OUTPUT    | LVCMOS25  |            |            |
+---------------------+-----------+-----------+------------+------------+
| data_out[3]         | OUTPUT    | LVCMOS25  |            |            |
+---------------------+-----------+-----------+------------+------------+
| data_out[2]         | OUTPUT    | LVCMOS25  |            |            |
+---------------------+-----------+-----------+------------+------------+
| data_out[1]         | OUTPUT    | LVCMOS25  |            |            |

+---------------------+-----------+-----------+------------+------------+
| data_in[7]          | INPUT     | LVCMOS25  |            |            |
+---------------------+-----------+-----------+------------+------------+
| data_in[6]          | INPUT     | LVCMOS25  |            |            |
+---------------------+-----------+-----------+------------+------------+
| data_in[5]          | INPUT     | LVCMOS25  |            |            |
+---------------------+-----------+-----------+------------+------------+
| data_in[4]          | INPUT     | LVCMOS25  |            |            |
+---------------------+-----------+-----------+------------+------------+
| data_in[3]          | INPUT     | LVCMOS25  |            |            |
+---------------------+-----------+-----------+------------+------------+
| data_in[2]          | INPUT     | LVCMOS25  |            |            |
+---------------------+-----------+-----------+------------+------------+
| data_in[1]          | INPUT     | LVCMOS25  |            |            |
+---------------------+-----------+-----------+------------+------------+
| data_in[0]          | INPUT     | LVCMOS25  |            |            |
+---------------------+-----------+-----------+------------+------------+
| miso                | INPUT     | LVCMOS25  |            |            |
+---------------------+-----------+-----------+------------+------------+
| dord                | INPUT     | LVCMOS25  |            |            |
+---------------------+-----------+-----------+------------+------------+
| cpol                | INPUT     | LVCMOS25  |            |            |
+---------------------+-----------+-----------+------------+------------+
| send                | INPUT     | LVCMOS25  |            |            |
+---------------------+-----------+-----------+------------+------------+
| clk                 | INPUT     | LVCMOS25  |            |            |
+---------------------+-----------+-----------+------------+------------+
```

Removed logic

```
Block VCC undriven or does not drive anything - clipped.
Block GND undriven or does not drive anything - clipped.
Block u0/GND undriven or does not drive anything - clipped.
Block u0/VCC undriven or does not drive anything - clipped.
Block u1/GND undriven or does not drive anything - clipped.
Block u1/VCC undriven or does not drive anything - clipped.
Block u2/GND undriven or does not drive anything - clipped.
Block u2/VCC undriven or does not drive anything - clipped.
Signal rst_bar_c_i was merged into signal rst_bar_c
Signal u2/CN was merged into signal u1.present_state[4]
Signal VCC undriven or does not drive anything - clipped.
Block rst_bar_pad_RNI1JBB was optimized away.
Block u2/g_1_.CN was optimized away.
```

GSR Usage

```
---------

GSR Component:
   The local reset signal 'rst_bar_c' of the design has been inferred as Global
        Set Reset (GSR). The reset signal used for GSR control is 'rst_bar_c'.


     GSR Property:
   The design components with GSR property set to ENABLED will respond to global
        set reset while the components with GSR property set to DISABLED will

        not.
```

Run Time and Memory Usage

```
-------------------------

   Total CPU Time: 0 secs
   Total REAL Time: 0 secs
   Peak Memory Usage: 27 MB
```