Lab 9: SPI Transmitter and Receiver

Ishabul Haque 111598085
Ken Ejinkonye 112011486

Section 2
05/04/20

File: C:/My_Designs/lab09/lab09/src/spi_tx_shifter.vhd (/spi_tx_shifter_tb/UUT)

157

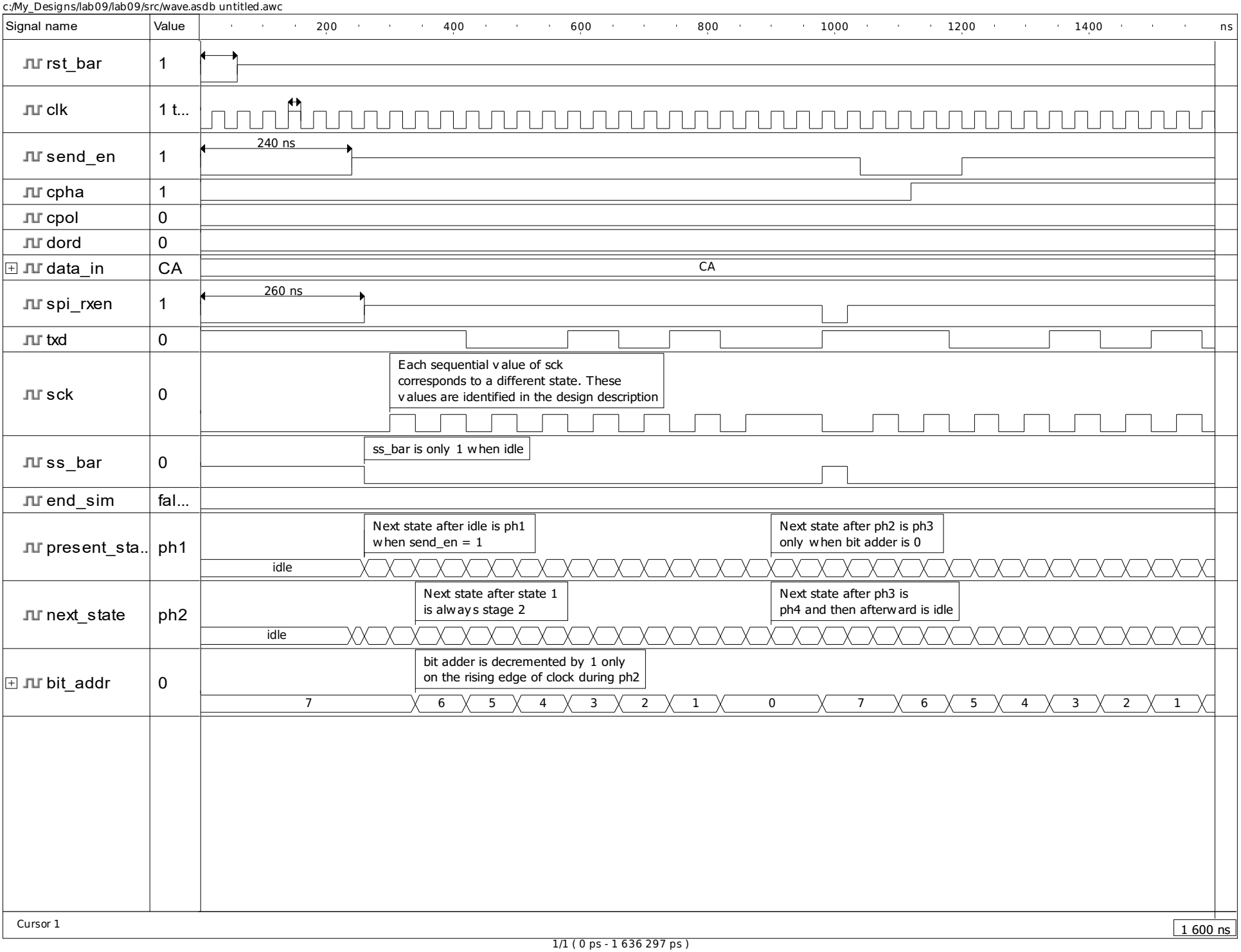
File: C:/My_Designs/lab09/lab09/src/spi_tx_shifter_tb.vhd (/spi_tx_shifter_tb)

```
1    ------------------------------------------------------------------------
     -----
2    --
3    -- Title       : spi_tx_shifter_tb
4    -- Design      : spi_tx_shifter_tb
5    -- Author      : Ishabul Haque and Ken Ejinkonye
6    -- Company     : Stony Brook
7    --
8    ------------------------------------------------------------------------
     -----
9    --
10   -- File        : c:\My_Designs\Lab_09\SPI_Transmitter_and_Reciever\src\spi_tx_shifter_t
     d
11   -- Generated   : Sun May  3 10:51:17 2020
12   -- From        : interface description file
13   -- By          : Itf2Vhdl ver. 1.22
14   --
15   ------------------------------------------------------------------------
     -----
16   --
17   -- Description : Non-self checking testbench for spi_tx_shifter design
18   --
19   ------------------------------------------------------------------------
     -----
20
21
22   library ieee;
23   use ieee.std_logic_1164.all;
24   use ieee.numeric_std.all;
25   library work;
26   use work.all;
27
28
29   entity spi_tx_shifter_tb is
30   end spi_tx_shifter_tb;
31
32
33
34   architecture tx_shifter_tb of spi_tx_shifter_tb is
35      --stimulus signals
36       signal rst_bar : std_logic;
37       signal clk : std_logic;
38       signal send_en: std_logic;
39       signal cpha: std_logic;
40       signal cpol: std_logic;
41       signal dord: std_logic;
42       signal data_in: std_logic_vector(7 downto 0);
43       signal spi_rxen: std_logic;
44
45       --observed signals
46       signal txd: std_logic;
47       signal sck: std_logic;
48       signal ss_bar: std_logic;
49
50
51       constant period : time := 40ns;
52       signal end_sim : boolean := false;
```

```
53
54   begin
55       -- Unit Under Test port map
56       UUT: entity spi_tx_shifter
57       port map (
58           rst_bar => rst_bar,
59           clk => clk,
60           send_en => send_en,
61           cpha => cpha,
62           cpol => cpol,
63           dord => dord,
64           data_in => data_in,
65           txd => txd,
66           sck => sck,
67           ss_bar => ss_bar,
68           spi_rxen => spi_rxen
69           );
70
71        -- Process to generate value for input data_in
72        data_gen:process
73        begin
74            for i in 1 to 2 loop
75                wait until clk='0';
76           data_in<="11001010";
77           end loop;
78           wait;
79        end process;
80
81        -- Process to generate clock
82
83        clock_gen:process
84        begin
85            clk<='0';
86            loop
87                wait for period/2;
88                clk<=not clk;
89                exit when end_sim = true;
90           end loop;
91           wait;
92        end process;
93
94       -- Process to generate values for reset bar
95       reset: process
96       begin
97           -- Reset bar intially 0 for 60ns
98           rst_bar <='0';
99           for i in 1 to 2 loop
100              wait until clk = '1';
101          end loop;
102          rst_bar<='1';
103          wait;
104      end process;
105
106      -- Process to generate values for dord,cpol,cpha. The push button
107      -- is represented by design send_pos_edge and cpol and cpha are
108      -- not changed in the middle of a transmission. Value of dord
109      -- determines which bit position the data is shifted to.
```

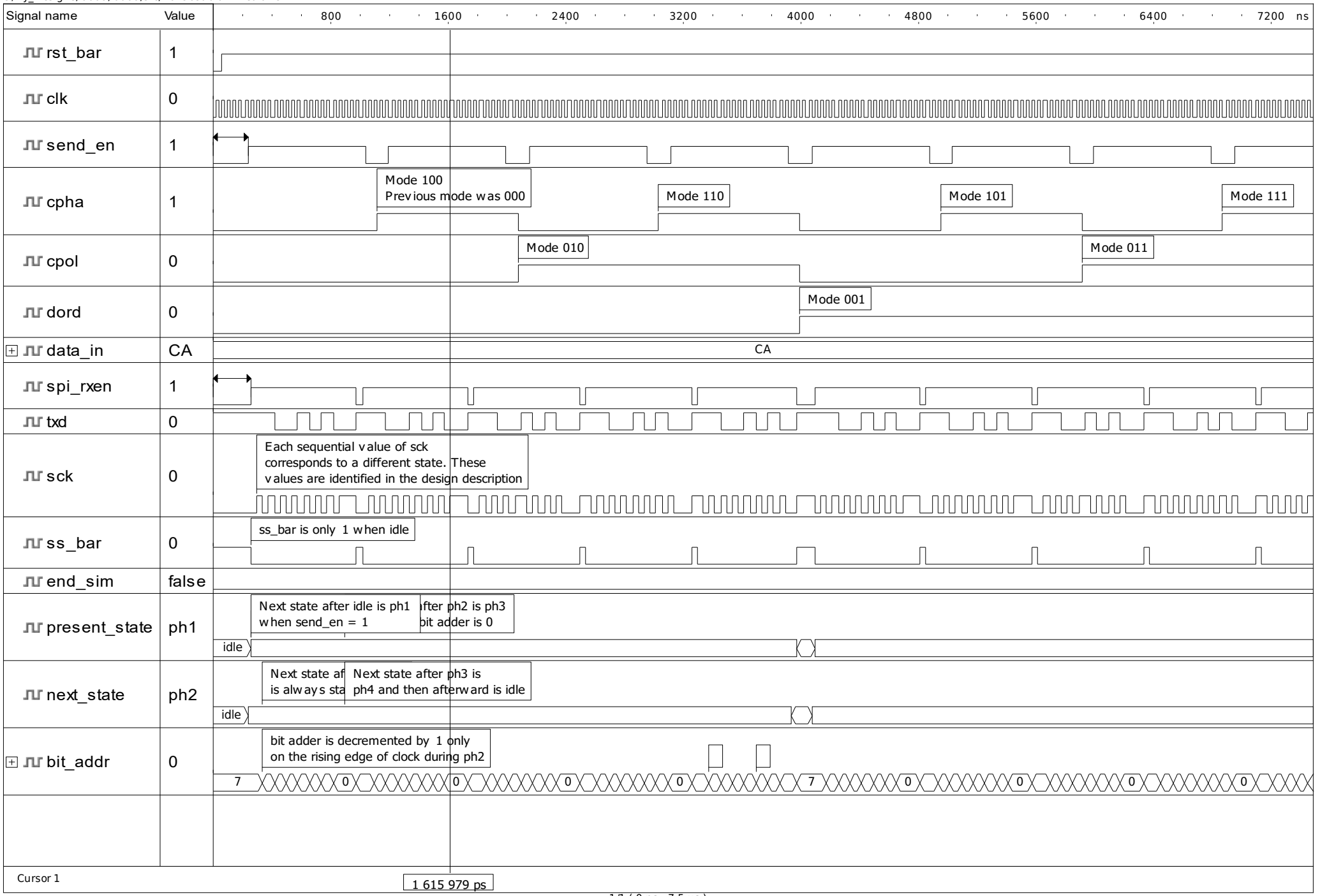
```
110      push_button:process
111      begin
112          -- Initialized values
113          send_en <= '0';
114          cpol <= '0';
115          cpha <= '0';
116          dord <= '0';
117
118          wait for 4*period;
119          for i in 0 to 7 loop
120              -- Generates the different modes of operation
121              (dord, cpol, cpha) <= to_unsigned(i,3);
122              wait for 2*period;
123              send_en <= '1';
124              wait for 20*period;
125              send_en <= '0';
126              wait for 2*period;
127
128          end loop;
129          end_sim <= true;
130          wait;
131      end process;
132
133
134  end tx_shifter_tb;
135
```

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



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

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



1/1 (0 ps - 7.5 us)

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

```
1   ----------------------------------------------------------------------------
    ----
2   --
3   -- Title       : spi_rx_shifter
4   -- Design      : task1_2
5   -- Author      : Ishabul Haque and Ken Ejinkonye
6   -- Company     : Stony Brook
7   --
8   -- Description : The receiver shifter spi_rx_shifter converts the serial
    data
9   -- at its rxd input to parallel and and provides this parallel result as
    output
10  -- data_out. This data may arrive most significant bit first or least
    significant
11  -- bit first, as determined by the dord input.The final parallel value at
    data_out
12  -- must always have its most significant bit in the leftmost position.
13
14  -- List of Circuit Features To Be Verified:
15  -- Inputs: rxd, rst_bar, clk,spi_rxen,dord
16
17  -- Outputs: data_out
18
19  ----------------------------------------------------------------------------
    ----
20
21  library ieee;
22  use ieee.std_logic_1164.all;
23  use ieee.numeric_std.all;
24
25
26  entity spi_rx_shifter is
27      port(
28      rxd : in std_logic;          -- data received from slave
29      rst_bar : in std_logic;      -- asynchronous reset
30      clk : in std_logic;          -- system clock
31      spi_rxen : in std_logic;     -- signal to enable shift
32      dord : in std_logic;         -- data order bit
33      data_out : out std_logic_vector(7 downto 0) -- received data
34      );
35  end spi_rx_shifter;
36
37
38  --}} End of automatically maintained section
39
40  architecture rx_shifter of spi_rx_shifter is
41  -- Signal g is a temporary vector to manipulate value going to data_out
42  signal g : std_logic_vector(7 downto 0):= "00000000";
43
44
45
46
47  begin
48
49      dord_change: process(dord, rxd, spi_rxen,g)
50      begin
51          -- Data can only be shifted when spi_rxen is asserted
```

```
52          if spi_rxen = '1' then
53              -- If dord = 0, data is shifted to most significant bit
54              if (dord='0') then
55                  -- Data is only shifted at the rising edge of clock
56                  if rising_edge(clk) then
57                      -- Data being shifted in is the value of rxd
58                      -- Initial bit position will always equal rxd
59                      data_out(0) <= rxd;
60                      g(0) <= rxd;
61                      -- For loop to implement shifting method using g
62                      -- as a test vector to manipulate data
63                      for i in 7 downto 0 loop
64                          if i>0 then
65                              g(i)<=g(i-1);
66                              data_out<=g;
67                          end if;
68
69
70                      end loop;
71                  end if ;
72              -- If dord = 1, data is shifted to least significant bit
73              elsif (dord='1') then
74                  if rising_edge(clk) then
75
76                      data_out(7)<=rxd;
77                      g(0) <= rxd;
78                      data_out <= g;
79
80                      for i in 7 downto 0 loop
81
82                          if i>0 then
83                              g(i-1)<=g(i);
84                              data_out<=g;
85                          end if;
86
87                      end loop;
88                  end if;
89
90
91              end if;
92          end if;
93
94      end process;
95
96  end rx_shifter;
97
98
```

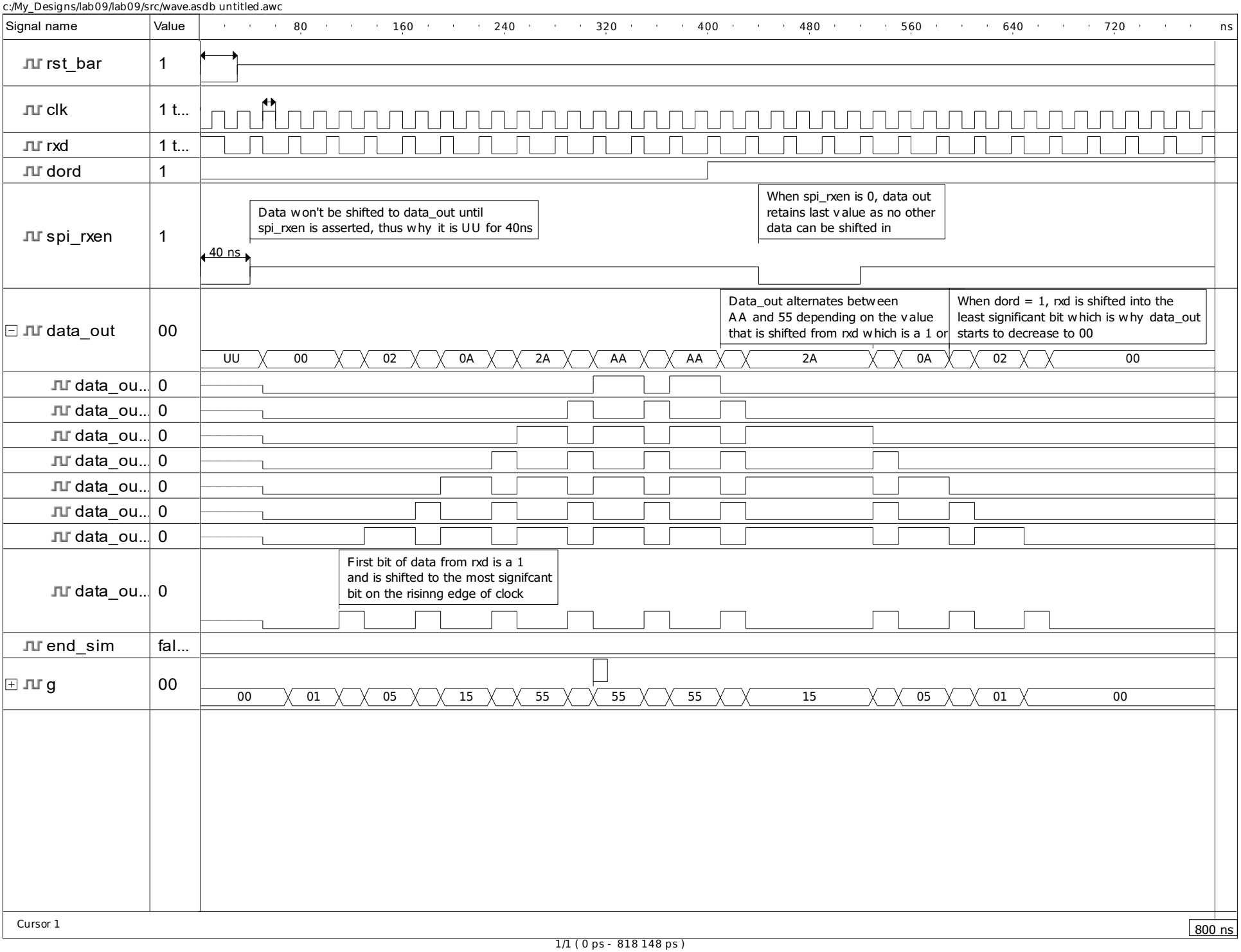
File: C:/My_Designs/lab09/lab09/src/spi_rx_shifter_tb.vhd

```
1    ------------------------------------------------------------------------
     -----
2    --
3    -- Title       : spi_rx_shifter_tb
4    -- Design      : spi_rx_shifter_tb
5    -- Author      : Ishabul Haque and Ken Ejinkonye
6    -- Company     : Stony Brook University
7    --
8    ------------------------------------------------------------------------
     -----
9    --
10   -- File        : c:\My_Designs\Lab_09\SPI_Transmitter_and_Reciever\src\spi_rx_shifter_t
     d
11   -- Generated   : Tue May  5 10:55:14 2020
12   -- From        : interface description file
13   -- By          : Itf2Vhdl ver. 1.22
14   --
15   ------------------------------------------------------------------------
     -----
16   --
17   -- Description : Non self checking test bench for spi_rx_shifter design
18   --
19   ------------------------------------------------------------------------
     -----
20
21
22   library ieee;
23   use ieee.std_logic_1164.all;
24   use ieee.numeric_std.all;
25
26
27
28   entity spi_rx_shifter_tb is
29   end spi_rx_shifter_tb;
30
31
32
33   architecture rx_shifter_tb of spi_rx_shifter_tb is
34
35       --stimulus signals
36       signal rst_bar : std_logic;
37       signal clk : std_logic;
38       signal rxd : std_logic;
39       signal dord: std_logic;
40       signal spi_rxen: std_logic;
41
42       --observed signals
43       signal data_out: std_logic_vector(7 downto 0);
44
45       -- Constants
46       constant period : time := 20ns;
47       signal end_sim : boolean := false;
48   begin
49       -- Unit Under Test port map
50       UUT: entity spi_rx_shifter
51       port map (
52           rst_bar => rst_bar,
```

```
53          clk => clk,
54          rxd => rxd,
55          dord => dord,
56          data_out => data_out,
57          spi_rxen => spi_rxen
58          );
59
60          -- Process to generate value for clock
61          clock_gen:process
62
63          begin
64
65              clk<='0';
66              loop
67                  wait for period/2;
68                  clk<=not clk;
69                  exit when end_sim = true;
70              end loop;
71              wait;
72          end process;
73
74          -- Process to generate value for reset bar
75          reset: process
76          begin
77
78              rst_bar<='0';
79              for i in 1 to 2 loop
80                  wait until clk = '1';
81
82              end loop;
83              rst_bar<='1';
84              wait;
85
86          end process;
87
88          -- Process to generate dord, dord simply changes
89          -- from 0 to 1 for verification purposes of the design
90          dord_gen: process
91          begin
92              loop
93              dord <= '0';
94              wait for 400 ns;
95              dord <= '1';
96              exit when end_sim = true;
97              end loop;
98          end process;
99
100         -- Process to generate value for spi_rxen
101         -- A value of 0 is asserted to verify data
102         -- is only shifted when spi_rxen is asserted
103         spi_rxen_gen: process
104
105         begin
106
107             spi_rxen <= '0';
108
109             loop
```

```
110                 wait for 2*period;
111                 spi_rxen <= '1';
112                 wait for 20*period;
113                 spi_rxen <= '0';
114                 wait for 2*period;
115                 exit when end_sim = true;
116             end loop;
117 
118         end process;
119 
120         -- Proocess to generate value for rxd, data
121         -- being shifted in
122         rxd_gen : process
123         begin
124             rxd <= '1';
125             loop
126                 wait for period/2;
127                 rxd <= '1';
128                 wait for period/2 ;
129                 rxd <= '0';
130                 wait for period/2;
131                 exit when end_sim = true;
132             end loop;
133         end process;
134 
135 
136 
137 end rx_shifter_tb;
138 
```

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



1/1 (0 ps - 818 148 ps)

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

```
1   ------------------------------------------------------------------------
    ----
2   --
3   -- Title       : send_pos_edge_det
4   -- Design      : send_pos_edge_det
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
```