Lab 8: Simple SPI Transmitter

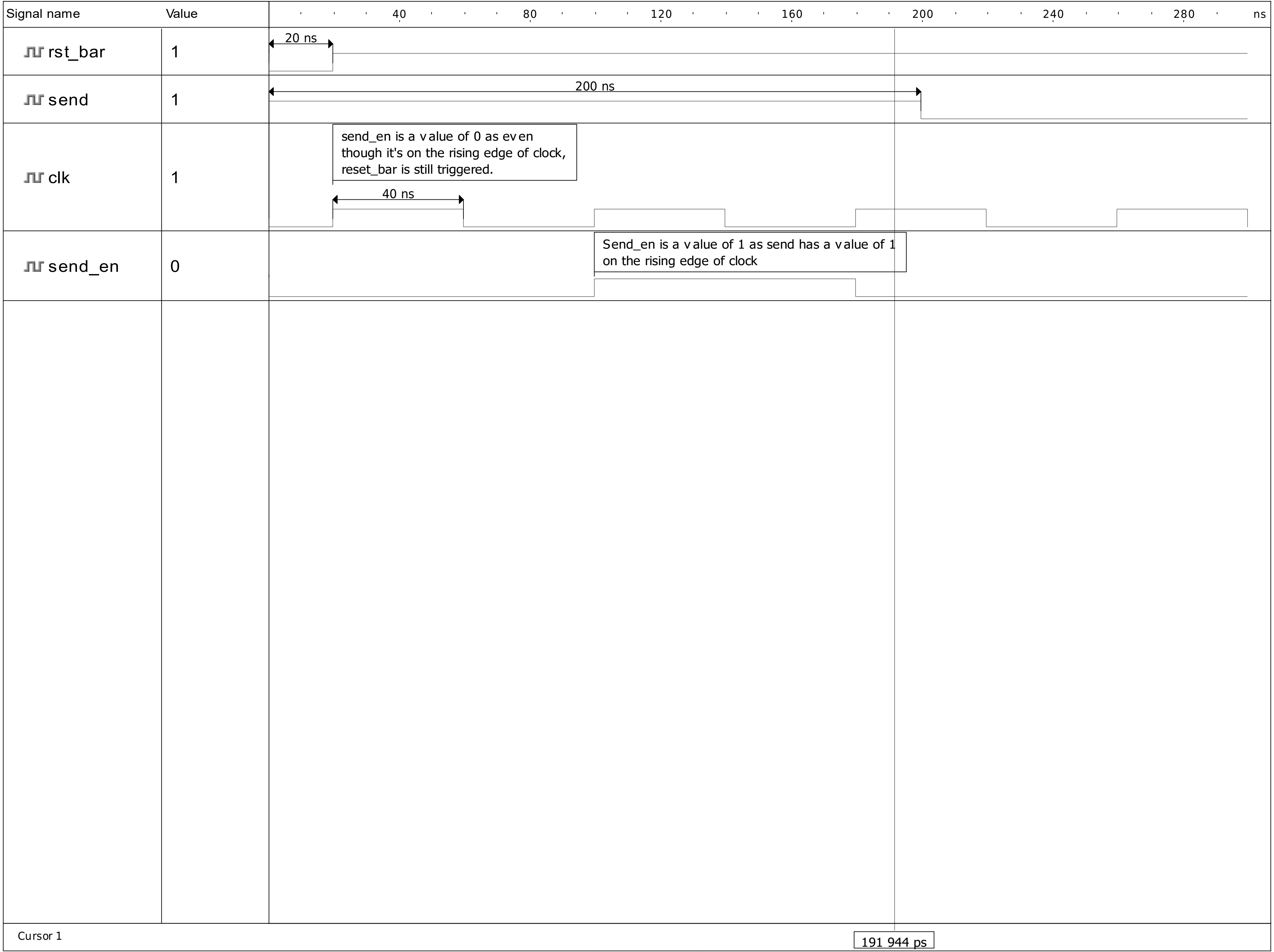
Ishabul Haque 111598085
Ken Ejinkonye 112011486

Section 2
04/10/20

```vhdl
--------------------------------------------------------------------------------
--
-- Title       : send_pos_edge_det
-- Design      : task1_2
-- 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       : send_pos_edge_det_tb
-- Design      : task1_2
-- Author      : Ishabul Haque and Ken Ejinkonye
-- Company     : Stony Brook
--
-- Description : Non self checking testbench for send_pos_edge_det design.
--
-------------------------------------------------------------------------------


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



entity send_pos_edge_det_TB is


end send_pos_edge_det_TB;

architecture tb_architecture of send_pos_edge_det_TB is


       --stimulus signals
    signal rst_bar : std_logic;
    signal send : std_logic;
    signal clk : std_logic;
        --observed signals
    signal send_en :  std_logic;

    constant period : time := 20ns; --need a much longer period
                                    -- in actual hardware

begin
    -- Unit Under Test port map
    UUT: entity send_pos_edge_det
    port map (
        clk => clk,
        rst_bar => rst_bar,
        send => send,
        send_en => send_en
        );

    rst_bar <= '0', '1' after period;   -- reset

    clock: process                      -- system clock
    begin
        -- clock starts at 0 for 0.5 clock periods
        for i in 0 to 28 loop
            wait for period;
            clk <= not clk;             -- 28 rising edges
            wait for period;
        end loop;                       -- stop clock
        std.env.finish;
    end process;

    -- Duration of signal send is 200ns at a high value
    snd: process
    begin
        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;

end tb_architecture;

```

C:/My_Designs/Lab8/lab8/task1/src/task1.asdb C:/My_Designs/Lab8/lab8/task1/src/task1.awc



Signal name	Value
rst_bar	1
send	1
clk	1
send_en	0

send_en is a value of 0 as even though it's on the rising edge of clock, reset_bar is still triggered.

Send_en is a value of 1 as send has a value of 1 on the rising edge of clock

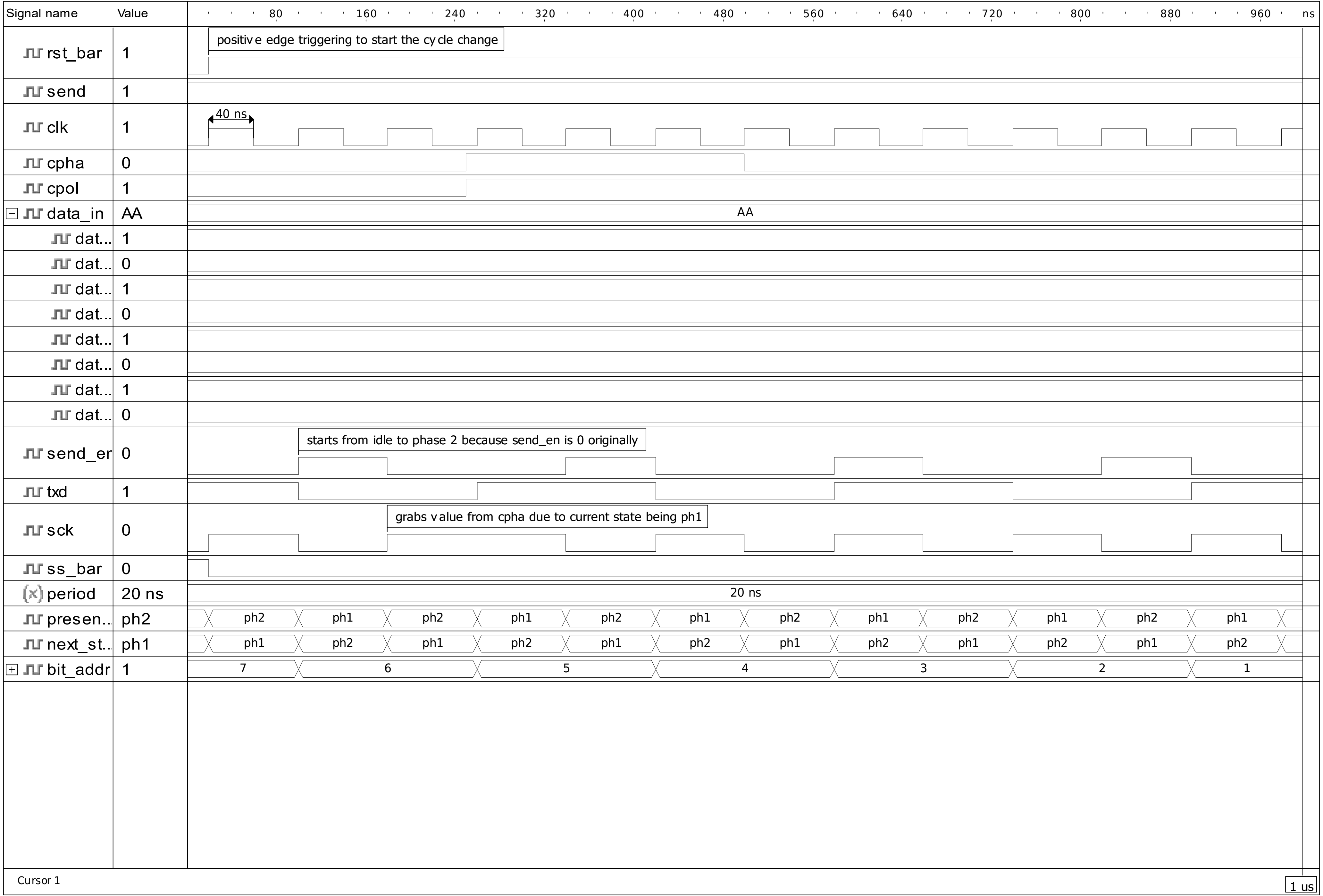
1/1 (0 ps - 307 235 ps)

```
-------------------------------------------------------------------------------
--
-- Title       : spi_tx_shifter
-- Design      : task1_2
-- Author      : Ishabul Haque and Ken Ejinkonye
-- Company     : Stony Brook
--
-- Description : The transmitter shifter spi_tx_shifter converts the parallel
-- data byte data_in to serial and transmits its data bits along with a
-- synchronous clock sck. An SPI slave uses sck to determine when to sample
-- each serial data bit. We will be transmitting the data's most significant
-- bit.



-- List of Circuit Features To Be Verified:
-- Inputs: rst_bar, clk, send_en, cpha,cpool,data_in
-- Inputs cpol and cpha determine the clock polarity and clock phase of the
-- transmitted dat.cpol determines whether the clock idles at 0 or 1. cpha
-- determines whether the data must be sampled on the leading or trailing
-- edge of the shift clock.The values of cpol and cpha are set to be
-- compatible with the SPI slave that receives the data.

-- Outputs: txd,sck, ss_bar
-- Output sck is used as the sample clock by the slave.

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

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

entity spi_tx_shifter is
    port(
    rst_bar : in std_logic;     -- asynnchronous system reset
    clk : in std_logic;         -- system clock
    send_en : in std_logic;         -- enable data transmission
    cpha : in std_logic;        -- clock phase
    cpol : in std_logic;        -- clock polarity
    data_in : in std_logic_vector(7 downto 0); -- data to send
    txd: out std_logic;         -- serial output data
    sck: out std_logic;         -- synchronous shift clock
    ss_bar: out std_logic       -- slave select
    );

end spi_tx_shifter;

architecture fsm of spi_tx_shifter is
```

```


type state is (idle, ph1, ph2);
signal present_state, next_state: state;
signal bit_addr : unsigned(2 downto 0);

begin


    state_reg: process(clk, rst_bar)
    begin
        -- Present State will be idle when
        -- reset bar is triggered
        if rst_bar = '0' then
            present_state <= idle;
        -- Present state will change to the
        -- next state only on the rising
        -- edge of clock
        elsif rising_edge(clk) then
            present_state <= next_state;
        end if;
    end process;

    -- Only when send_en is high is when next state goes
    -- to phase 1, otherwise it goes to phase 2 from idle
    -- When bit adder is 000, it goes back to idle.
    nxt_state: process (present_state, send_en, bit_addr)
    begin
        case present_state is
            when idle =>
            --
            if send_en = '1' then
                next_state <= ph1;
            else
                next_state <= ph2;
            end if;

            when ph1 =>
            next_state <= ph2;

            when ph2 =>
            if bit_addr = "000" then
                next_state <= idle;
            else
                next_state <= ph1;
            end if;
        end case;
    end process;

    -- Values to be assigned to the output signals
    -- based on the current state and value of
    -- bit adder
    output: process (present_state, data_in, bit_addr)
    begin
        case present_state is

            when idle=>
            sck <= cpol;
            ss_bar <= '1';
            txd <= data_in(to_integer(bit_addr));
            when ph1=>
            sck<=cpol;
            ss_bar<='0';
            txd<= data_in(to_integer(bit_addr));
            when ph2=>
            sck <= not cpol;
            ss_bar <= '0';
            txd <= data_in(to_integer(bit_addr));
        end case;
    end process;

    -- Bit counter is used to implement shifting by using count to
    -- select which bit from the parallel input data is output
    bit_counter: process (rst_bar, clk, present_state)
    begin
        if rst_bar = '0' or present_state = idle then
            bit_addr <= "111";
        elsif rising_edge(clk) then
            if present_state = ph2 then
                if bit_addr /= "000" then
                    bit_addr <= bit_addr - 1;
                end if;
            end if;
        end if;
    end process;


end fsm;
```

```vhdl
-------------------------------------------------------------------------------
--
-- Title       : task2_tb
-- Design      : task1_2
-- Author      : Ishabul Haque and Ken Ejinkonye
-- Company     : Stony Brook
--
-- Description : Non-self checking testbench for spi_tx_shifter
-------------------------------------------------------------------------------


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


entity task2_tb is
    -- Generic declarations of the tested unit




--}} End of automatically maintained section
end task2_tb;

architecture tb_architecture of task2_tb is


       --stimulus signals
    signal rst_bar : std_logic;
    signal send : std_logic := '1';
    signal clk : std_logic := '0';
    signal cpha: std_logic :='0';
    signal cpol: std_logic := '0';
    signal data_in: std_logic_vector(7 downto 0);

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


    constant period : time := 20ns; --need a much longer period
                                    -- in actual hardware

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,
        data_in => data_in,
        txd => txd,
        sck => sck,
        ss_bar => ss_bar
        );

    -- Port mapping for send_en from send_pos_edge_det
    -- design to spi_tx_shifter design
    u0 : entity send_pos_edge_det
    port map(
    rst_bar => rst_bar, clk => clk,
    send_en => send_en,
    send=>send);
    rst_bar <= '0', '1' after period * 3;   -- reset

    clock: process                      -- system clock
    begin
        -- clock starts at 0 for 0.5 clock periods
        for i in 0 to 28 loop
            wait for period;
            clk <= not clk;             -- 28 rising edges
            wait for period;
        end loop;                       -- stop clock
        std.env.finish;
    end process;




end tb_architecture;

```

c:/My_Designs/Lab_08/lab_08/src/test002.asdb c:/My_Designs/Lab_08/lab_08/src/test002.awc



Signal name	Value
rst_bar	1
send	1
clk	1
cpha	0
cpol	1
data_in	AA
dat...	1
dat...	0
dat...	1
dat...	0
dat...	1
dat...	0
dat...	1
dat...	0
send_er	0
txd	1
sck	0
ss_bar	0
period	20 ns
presen...	ph2
next_st...	ph1
bit_addr	1

positive edge triggering to start the cycle change

starts from idle to phase 2 because send_en is 0 originally

grabs value from cpha due to current state being ph1

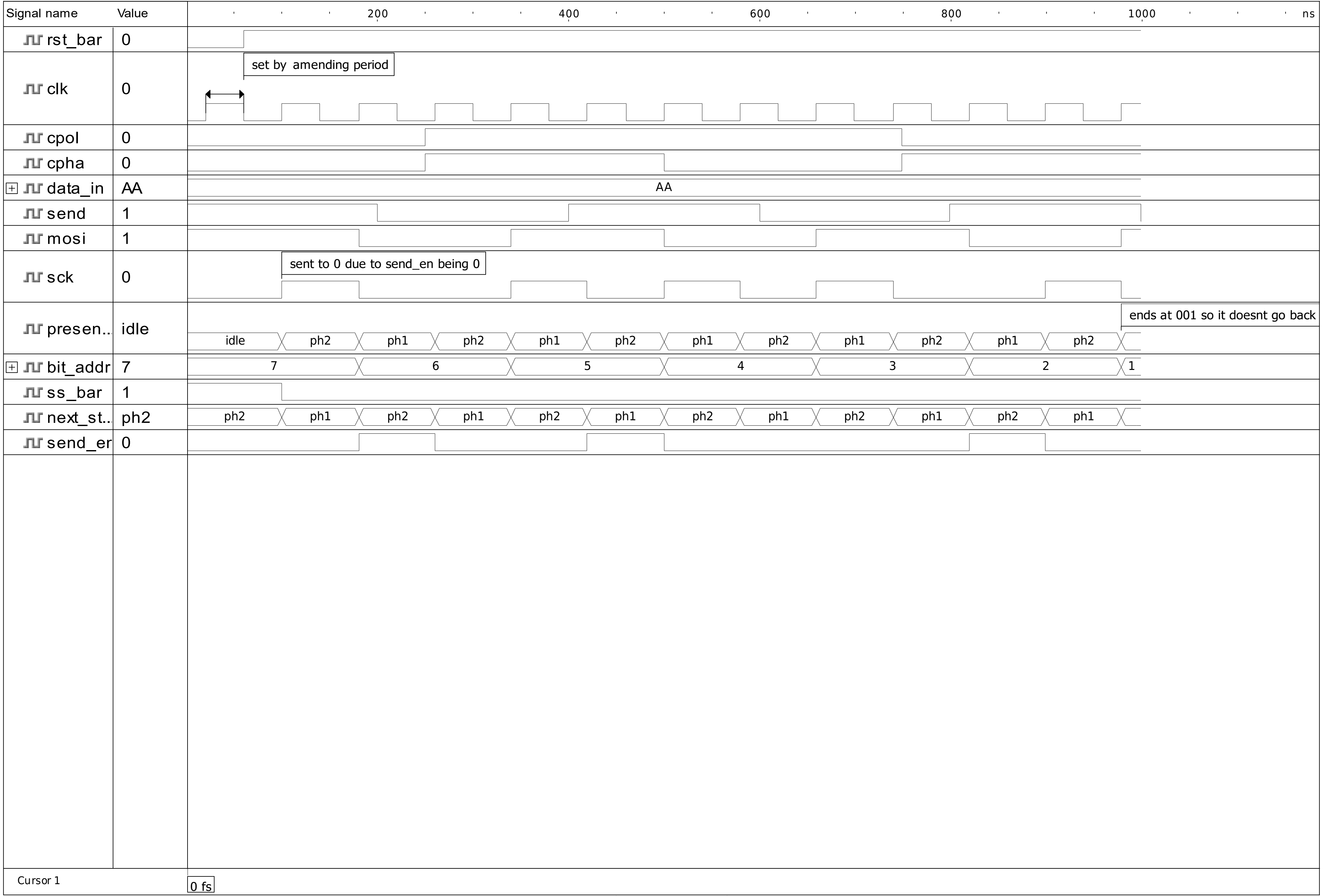
80
160
240
320
400
480
560
640
720
800
880
960
ns
40 ns
AA
20 ns
ph2 ph1 ph2 ph1 ph2 ph1 ph2 ph1 ph2 ph1 ph2 ph1
ph1 ph2 ph1 ph2 ph1 ph2 ph1 ph2 ph1 ph2 ph1 ph2
7 6 5 4 3 2 1
Cursor 1
1 us

1/1 (0 ps - 1 015 265 ps)

```vhdl
-------------------------------------------------------------------------------
--
-- Title       : SPI_test_system
-- Design      : task1_2
-- Author      : Ishabul Haque and Ken Ejinkonye
-- Company     : Stony Brook
--
-- Description : The top-level structure simply combines the two previous entities
-- send_pos_edge_det and spi_tx_shifter.
-- List of Circuit Features to be verified:
-- Inputs: rst_bar, clk, send, cpol, cpha, data_in
-- Outputs: mosi, sck, ss_bar
-- Observant Stimulus: Presennt State, Next State
-------------------------------------------------------------------------------

library IEEE;
use IEEE.std_logic_1164.all;
library work;
use work.all;

entity SPI_test_system_I is
    port(
    rst_bar : in std_logic;     -- asynnchronous system reset
    clk : in std_logic;         -- system clock
    send : in std_logic;        -- debounced send input
    cpol : in std_logic;        -- clock poolarity settinng
    cpha : in std_logic;        -- clock phase setting
    data_in : in std_logic_vector(7 downto 0);  -- parallel input 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      -- SPI shift clock to slave
    );
end SPI_test_system_I;

architecture structural of SPI_test_system_I is

signal temp_send_en : 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, data_in => data_in, sck => sck,
    ss_bar => ss_bar, send_en => temp_send_en, txd => mosi);




end structural;
```

```
-------------------------------------------------------------------------------
--
-- Title       : task3_tb
-- Design      : task1_2
-- Author      : Ishabul Haque and Ken Ejinkonye
-- Company     : Stony Brook
--
-- Description : Non self checking test bench for SPI_test_system
-------------------------------------------------------------------------------


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



entity task3_tb is
    -- Generic declarations of the tested unit




--}} End of automatically maintained section
end task3_tb;

architecture tb_architecture of task3_tb is


       --stimulus signals
    signal rst_bar : std_logic;
    signal send : std_logic := '1';
    signal cpha : std_logic;
    signal cpol: std_logic;
    signal clk : std_logic := '0';

    signal data_in: std_logic_vector(7 downto 0);

    --observed signals
    signal send_en: std_logic;
    signal txd: std_logic;
    signal sck: std_logic;
    signal ss_bar: std_logic;
    signal present_state:std_logic;
    signal next_state:std_logic;
    signal mosi : std_logic;

    constant period : time := 20ns; --need a much longer period
                                    -- in actual hardware

begin
    -- Unit Under Test port map

    u0 : entity send_pos_edge_det port map(rst_bar => rst_bar, clk => clk,
    send => send, send_en => send_en);

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

    rst_bar <= '0', '1' after period * 3;   -- reset

    clock: process              -- system clock
    begin
                    -- clock starts at 0 for 0.5 clock periods
        for i in 0 to 28 loop
            wait for period;
            clk <= not clk;     -- 15 rising edges
            wait for period;
        end loop;                   -- stop clock
        std.env.finish;
    end process;

    snd: process
    begin
        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;



end tb_architecture;
```

C:/My_Designs/Lab_08/lab_08/src/task3.asdb C:/My_Designs/Lab_08/lab_08/src/task3.awc



set by amending period

sent to 0 due to send_en being 0

ends at 001 so it doesnt go back

1/1 (0 ps - 1 187 959 ps)

1. How long is the send_en pulse generated by send_pos_edge_det? State this time duration as both number of clock pulses and as ns.
    a. The send_en pulse generated by the designs task is 80ns long and is the duration of two clock pulses
2. What is the frequency relationship between the SPI clock sck and the system clock clk?
    a. The frequency of SPI clock sck is half of system clock clk.
3. From the log file determine the default state assignment encoding used by the synthesizer for your FSMs.