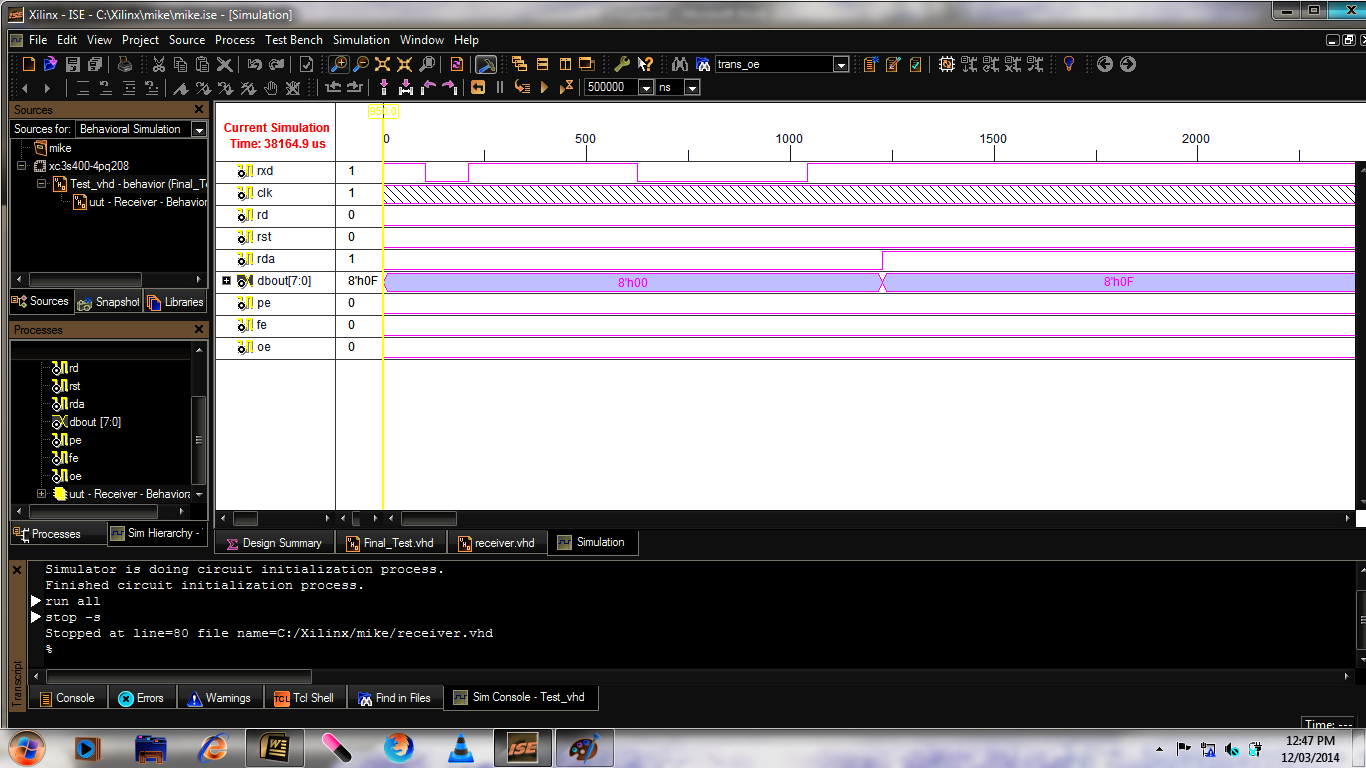
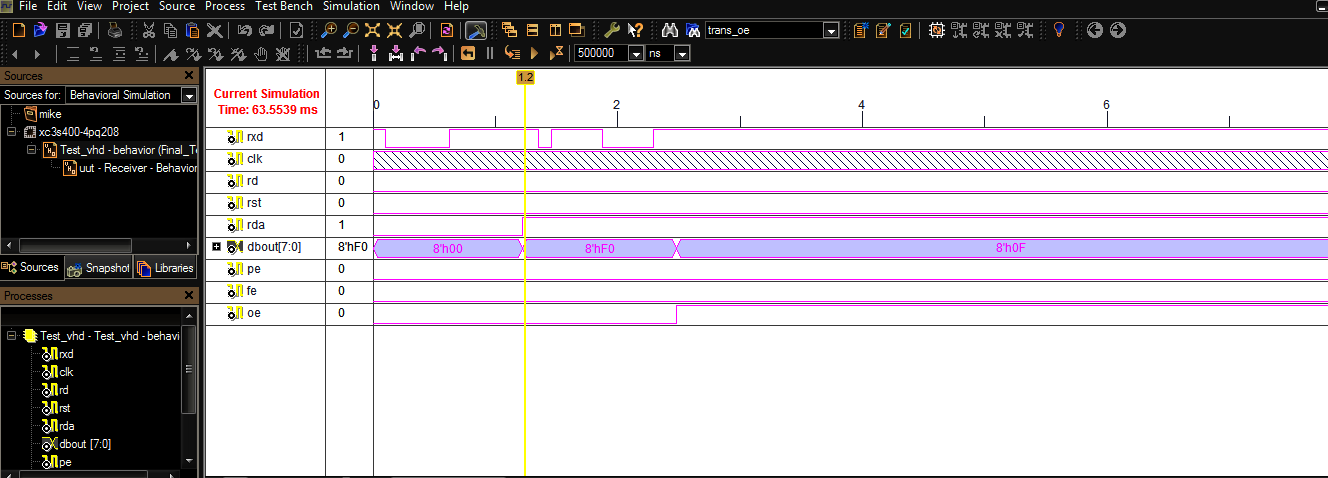
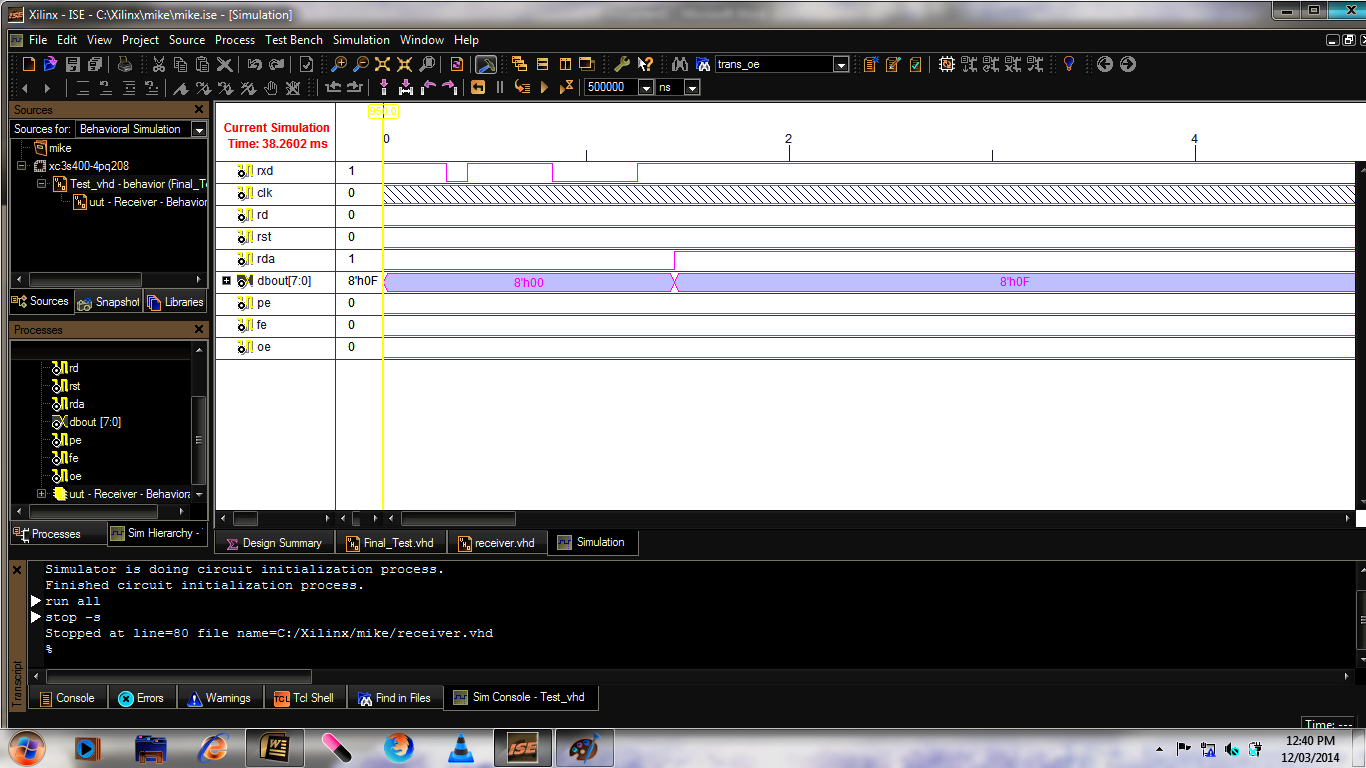
without error :



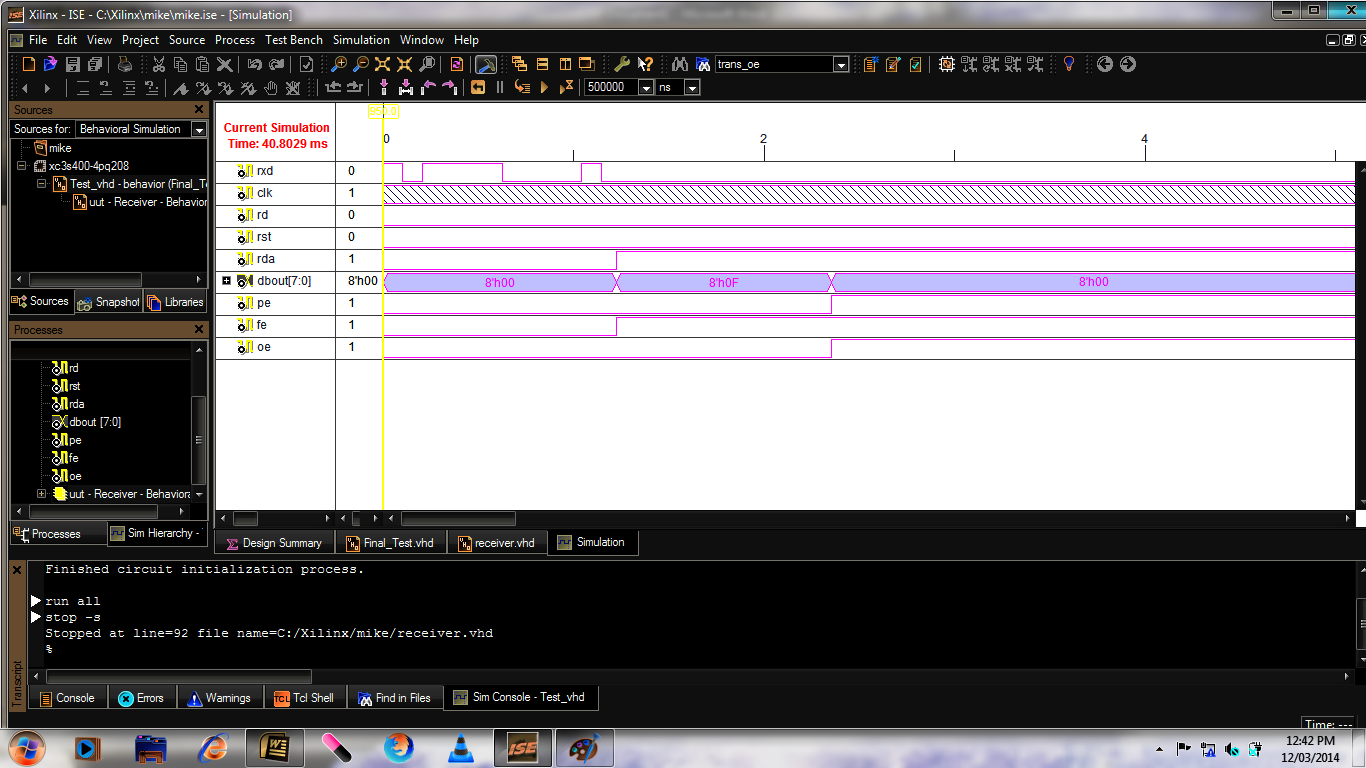
over write error:



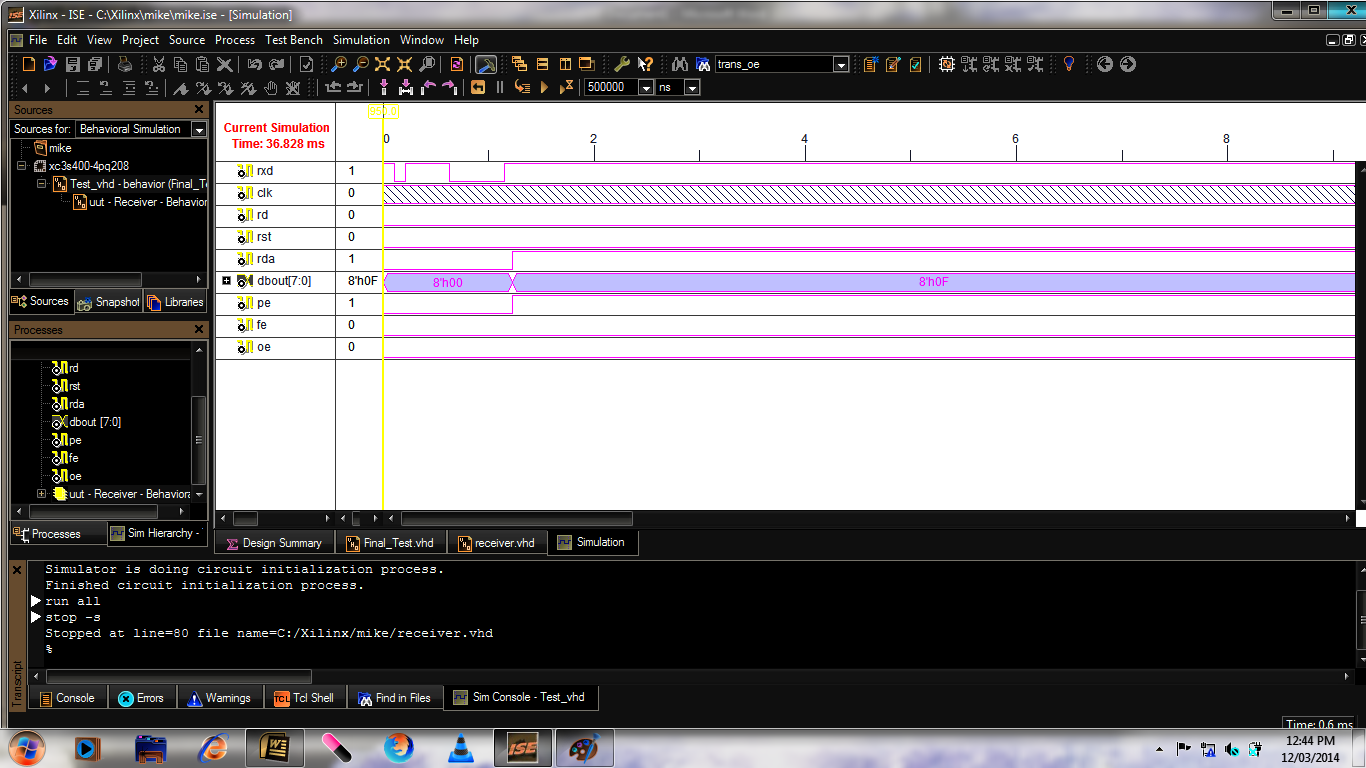
idle condition :



frame error :



parity error :



**code :**

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

-- Company:

-- Engineer:

--

-- Create Date: 16:40:23 03/09/2014

-- Design Name:

-- Module Name: receiver - Behavioral

-- Project Name:

-- Target Devices:

-- Tool versions:

-- Description:

--

-- Dependencies:

--

-- Revision:

-- Revision 0.01 - File Created

-- Additional Comments:

--

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

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity Receiver is

Port (

RXD : in std\_logic;

CLK : in std\_logic;

DBOUT : out std\_logic\_vector (7 downto 0);

RDA : inout std\_logic;

RD : in std\_logic;

PE : out std\_logic;

FE : out std\_logic;

OE : out std\_logic;

RST : in std\_logic);

end Receiver;

architecture Behavioral of Receiver is

type rstate is (

strIdle,

strEightDelay,

strGetData,

strCheckStop

);

constant baudDivide : std\_logic\_vector(3 downto 0) := "1101";

signal rdReg : std\_logic\_vector(7 downto 0) := "00000000";

signal rdSReg : std\_logic\_vector(9 downto 0) := "0000000000";

signal clkDiv : std\_logic\_vector(3 downto 0) := "0000";

signal rClkDiv : std\_logic\_vector(3 downto 0) := "0000";

signal ctr : std\_logic\_vector(3 downto 0) := "0000";

signal rClk : std\_logic := '1';

signal dataCtr : std\_logic\_vector(3 downto 0) := "0000";

signal parError: std\_logic;

signal frameError: std\_logic;

signal CE : std\_logic;

signal ctRst : std\_logic := '0';

signal rShift : std\_logic := '0';

signal dataRST : std\_logic := '0';

signal dataIncr: std\_logic := '0';

signal strCur : rstate := strIdle;

signal strNext : rstate;

begin

frameError <= not rdSReg(9);

parError <= not ( rdSReg(8) xor (((rdSReg(0) xor rdSReg(1)) xor (rdSReg(2) xor rdSReg(3))) xor ((rdSReg(4) xor rdSReg(5)) xor (rdSReg(6) xor rdSReg(7)))) );

--DBOUT <= rdReg;

process (CLK, clkDiv)

begin

if (Clk = '1' and Clk'event) then

if (clkDiv = baudDivide) then

rclk<= not rclk;

clkDiv <= "0000";

else

rclk<= rclk;

clkDiv <= clkDiv +1;

end if;

end if;

end process;

process (rClk)

begin

if (rClk = '1' and rClk'event) then

rClkDiv <= rClkDiv +1;

end if;

end process;

process(rclk,ctRst)

begin

if rClk = '1' and rClk'Event then

if ctRst = '1' then

ctr <= "0000";

else

ctr <= ctr +1;

end if;

end if;

end process;

process (rClk, RST, RD, CE)

begin

if RD = '1' or RST = '1' then

FE <= '0';

OE <= '0';

RDA <= '0';

PE <= '0';

elsif rClk = '1' and rClk'event then

if CE = '1' then

FE <= frameError;

OE <= RDA;

RDA <= '1';

PE <= parError;

rdReg(7 downto 0) <= rdSReg (7 downto 0);

end if;

end if;

Dbout<=rdReg;

end process;

process (rClk, rShift)

begin

if rClk = '1' and rClk'Event then

if rShift = '1' then

rdSReg <= (RXD & rdSReg(9 downto 1));

end if;

end if;

end process;

process (rClk, dataRST)

begin

if (rClk = '1' and rClk'event) then

if dataRST = '1' then

dataCtr <= "0000";

elsif dataIncr = '1' then

dataCtr <= dataCtr +1;

end if;

end if;

end process;

process (rClk, RST)

begin

if rClk = '1' and rClk'Event then

if RST = '1' then

strCur <= strIdle;

else

strCur <= strNext;

end if;

end if;

end process;

process (strCur, ctr, RXD, dataCtr, rdSReg, rdReg, RDA)

begin

case strCur is

when strIdle =>

dataIncr <= '0';

rShift <= '0';

dataRst <= '0';

CE <= '0';

if RXD = '0' then

ctRst <= '1';

strNext <= strEightDelay;

else

ctRst <= '0';

strNext <= strIdle;

end if;

when strEightDelay =>

dataIncr <= '0';

rShift <= '0';

CE <= '0';

if ctr(2 downto 0) = "111" then

ctRst <= '1';

dataRST <= '1';

strNext <= strGetData;

else

ctRst <= '0';

dataRST <= '0';

strNext <= strEightDelay;

end if;

when strGetData =>

CE <= '0';

dataRst <= '0';

if ctr(3 downto 0) = "1110" then

ctRst <= '1';

dataIncr <= '1';

rShift <= '1';

else

ctRst <= '0';

dataIncr <= '0';

rShift <= '0';

end if;

if dataCtr = "1010" then

strNext <= strCheckStop;

else

strNext <= strGetData;

end if;

when strCheckStop =>

dataIncr <= '0';

rShift <= '0';

dataRst <= '0';

ctRst <= '0';

CE <= '1';

strNext <= strIdle;

end case;

end process;

end Behavioral;

**testbench :**

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

-- Company:

-- Engineer:

--

-- Create Date: 22:47:02 03/09/2014

-- Design Name: Receiver

-- Module Name: C:/Xilinx/mike/tes2t.vhd

-- Project Name: mike

-- Target Device:

-- Tool versions:

-- Description:

--

-- VHDL Test Bench Created by ISE for module: Receiver

--

-- Dependencies:

--

-- Revision:

-- Revision 0.01 - File Created

-- Additional Comments:

--

-- Notes:

-- This testbench has been automatically generated using types std\_logic and

-- std\_logic\_vector for the ports of the unit under test. Xilinx recommends

-- that these types always be used for the top-level I/O of a design in order

-- to guarantee that the testbench will bind correctly to the post-implementation

-- simulation model.

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

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

USE ieee.std\_logic\_unsigned.all;

USE ieee.numeric\_std.ALL;

ENTITY Test\_vhd IS

END Test\_vhd;

ARCHITECTURE behavior OF Test\_vhd IS

-- Component Declaration for the Unit Under Test (UUT)

COMPONENT Receiver

PORT(

RXD : IN std\_logic;

CLK : IN std\_logic;

RD : IN std\_logic;

RST : IN std\_logic;

RDA : INOUT std\_logic;

DBOUT : OUT std\_logic\_vector(7 downto 0);

PE : OUT std\_logic;

FE : OUT std\_logic;

OE : OUT std\_logic

);

END COMPONENT;

--Inputs

SIGNAL RXD : std\_logic:='0';

SIGNAL CLK : std\_logic := '0';

SIGNAL RD : std\_logic := '0';

SIGNAL RST : std\_logic := '0';

--BiDirs

SIGNAL RDA : std\_logic;

--Outputs

SIGNAL DBOUT : std\_logic\_vector(7 downto 0);

SIGNAL PE : std\_logic;

SIGNAL FE : std\_logic;

SIGNAL OE : std\_logic;

constant Period: time:=250 ns;

constant BitPeriod: time:=104166 ns;

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: Receiver PORT MAP(

RXD => RXD,

CLK => CLK,

DBOUT => DBOUT,

RDA => RDA,

RD => RD,

PE => PE,

FE => FE,

OE => OE,

RST => RST

);

process

begin

clk<='0';

wait for Period/2;

clk<='1';

wait for Period/2;

end process;

rst<='1','0' after Period/2;

tb : PROCESS

variable v:std\_logic;

variable cnt:integer:=0;

--data="00001111"

constant data:std\_logic\_vector(11 downto 0):="110000111101"; -- without error

--constant data:std\_logic\_vector(11 downto 0):="100000111101"; -- to check parity error

--constant data:std\_logic\_vector(11 downto 0):="010000111101"; -- to check frame error

--constant data:std\_logic\_vector(13 downto 0):="11000011110111"; --to check iddle condition

--constant data:std\_logic\_vector(23 downto 0):="110000111101111111000001"; -- overwrite error with data F0 & 0F

BEGIN

if(cnt<12) then

rxd<=data(cnt);

wait for BitPeriod;

end if;

--wait until clk'event and clk='1';

--cnt:= cnt+1;

wait until clk'event and clk='1';

cnt:= cnt+1;

--v:=data(cnt);

--cnt:= cnt+1;

END PROCESS;

END;