UART Receiver – Project Report

EC-104: Digital Logic Design

Project Submitted By:

Sarjan Jain : 13116034
Kandikunta Achut : 13116035
Kartik Patel : 13116036
K. Nagadeep : 13116037
Kumar Parimal : 13116038

Class: B. Tech. ECE 1st Year

PROJECT STATEMENT

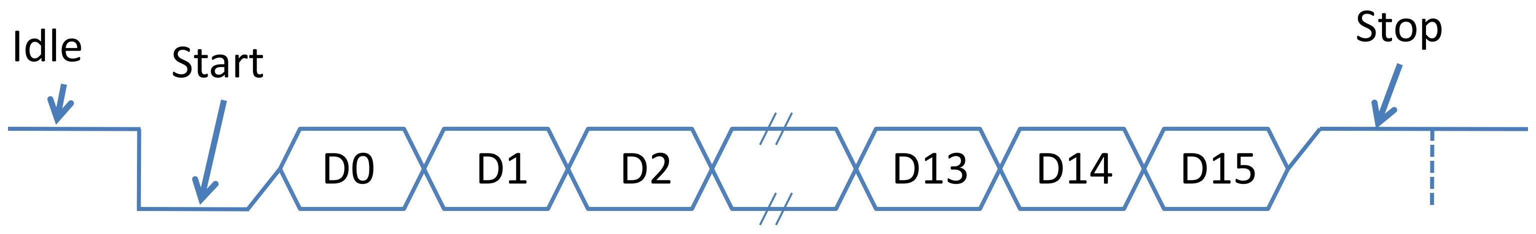
It converts serial bytes it receives into parallel data for outbound transmission.

Design a circuit to take 16 bit serial data as input and output it in parallel form.

INDEX

Introduction

- Takes 16-bit serial Input and convert it to parallel 16-bit form
- For transmission, UART protocol wraps 16-bit data with start and stop bit
- Optional parity bit is included just before stop bit
- For reception, first start bit is detected and based on the input clock serial data is sampled and converted to parallel form.



Basic Concept

- When data is ready, transmitter sends “Start bit” to alert Receiver
- After that individual bits of the data is sent in serial with LSB first
- Receiver continuously receives serial bits with clock and convert them in parallel form
- When all data is sent a stop bit is sent by transmitter thereby signalling receiver to remain in “Start Bit Detection” state
- Optional parity check is also possible by adding parity bit just before last bit

Basic Concept(Contd.)

- Start bit detection and validation
  - High to Low transition indicates a start bit.
  - Start bit validated if Receiver input is low during mid bit sampling
- A valid stop bit is HIGH when the stop bit is sampled

Approach

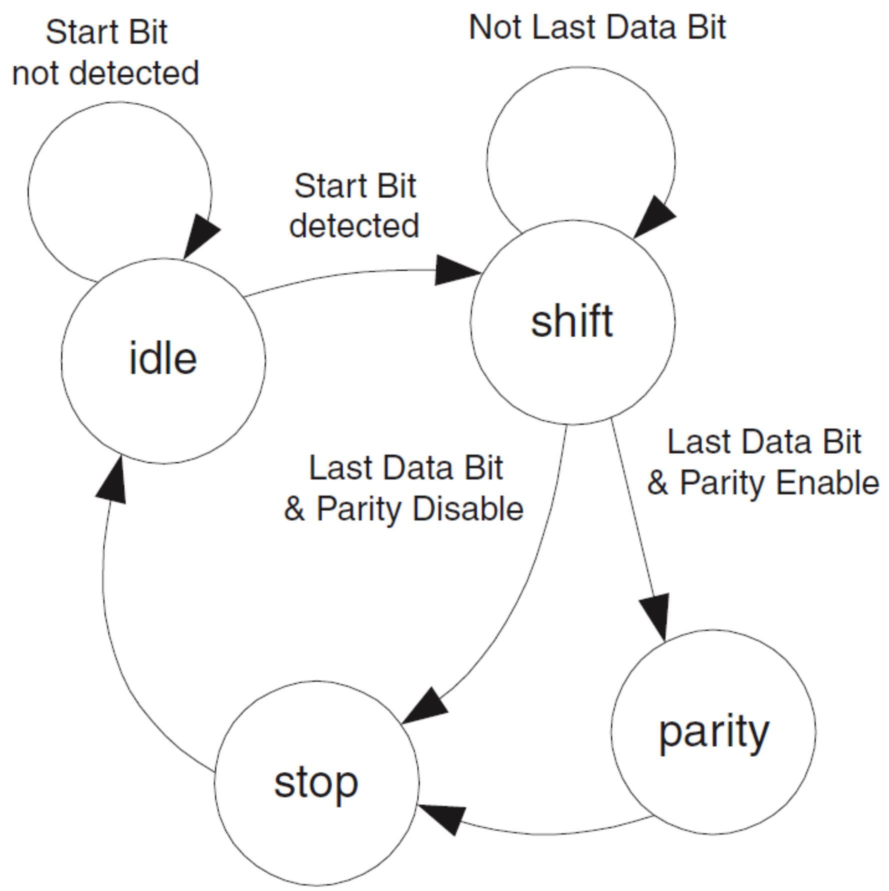
- Assumption made:
    - Clock input is available which is 16 times defined baud rate
- First, Finite State Machine (FSM) diagrams are prepared before designing a logic in VHDL
- VHDL code is written and checked for syntax error
- Simulation carried out.

Approach(Contd.)

- We have made Entity named CLK16 which converts given CLK input to 16 times CLK Rate. Here CLK input is in fact output of Baud Generator.
- We have made Entity named UART_Receiver with 6 ports with 3 input and 3 Output Ports. One of the Output port is 16-bit data bus which is our parallel output.
- At first receiver needs Start Bit. After successful sampling of Start Bit, receiver will sample every further inputs with counting up to 16.
- At 17th count, receiver takes SER_IN as Parity bit and checks for validation.
- If validated than it will transmit data to data bus. Else it will transmit ‘0’ with Error=‘1’.

Design

FSM Diagram



Idle:

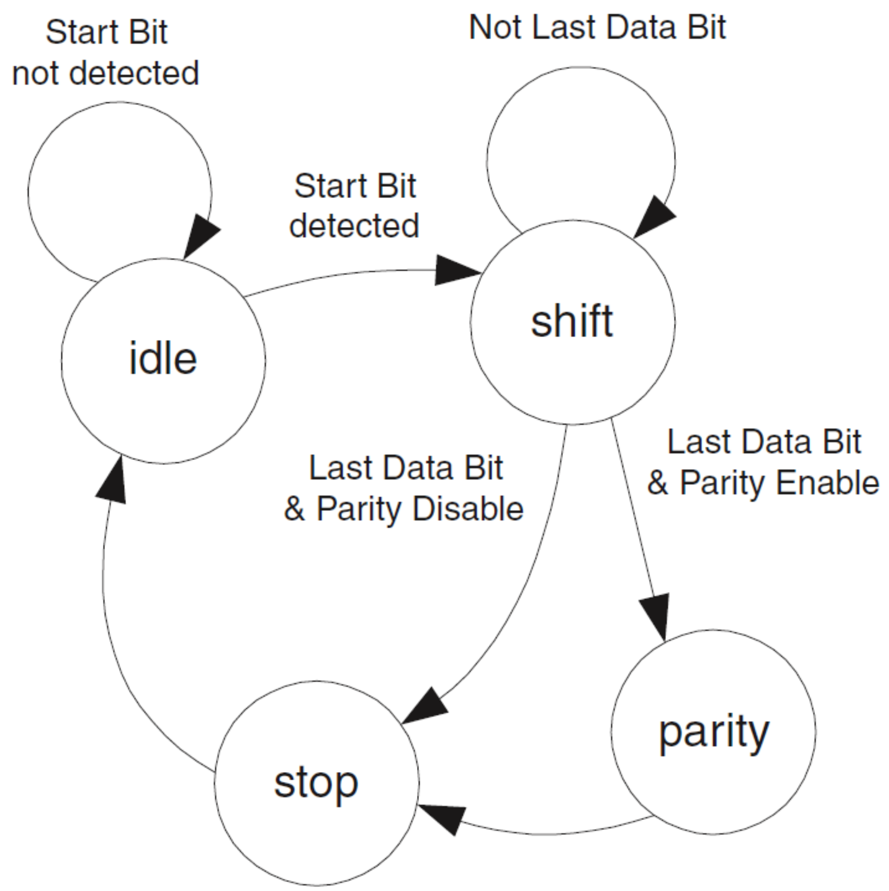
- Reset Button will reset machine to this state.
- In this state, receiver is waiting for SER_IN to go from high to low i.e. start bit. After successful detection, it will shift to <shift> state.

Shift:

- When Receiver is in this state, it waits for 1 clock cycle for each data bit to shift.
- After last Data bit is shifted in the receiver will switch to <parity> state if parity is enabled.
- Otherwise it will switch to <stop> state

Design

FSM Diagram



Parity:

- When the FSM is in this state, it waits for next clock cycle and then samples parity bit.
- Once it is sampled it will switch to <stop> set.

Stop:

- FSM will wait for 1 clock cycle and then samples the stop bit.
- The FSM switches back to <idle> state after the Stop bit sampling

VHDL Code

Entity Declaration for CLK16

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity CLK16 is
port (CLK : in std_logic;
      enable: in std_logic:='0';
      CLK16x : out std_logic := '0');
end CLK16;

architecture Behavioral of CLK16 is
signal currentCLK16x : std_logic :='1';
signal count : integer := 16;
```

Ports Required:

- CLK: Clock input
- CLK16x : Output Port which changes values at every 16 counts. This is input for Receiver

VHDL Code

Architecture Block of CLK16

```
begin
process(CLK, enable)
begin
  if enable=‘1’ then
    if Rising_edge(CLK) then
      if count=16 then
        currentCLK16x <= not currentCLK16x;
        CLK16x<=currentCLK16x;
      end if;
      count <= count-1;
      if count=0 then
        count<=16;
      end if;
    end if;
  end if;
end process;
end Behavioral;
```

VHDL Code

Entity Declaration

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity UART_Receiver is
port(SER_IN : in std_logic := '1';
     RESET : in std_logic;
     OUT16O : out std_logic_vector(15 downto 0):=(others=>'0');
     CLK : in std_logic;
     statusOUT : out std_logic :='0';
     ERROR: out std_logic := '0');
end UART_Receiver;
```

Ports Required:

- CLK: Clock input for UART. Which is defined Baud rate.
- RESET: Asynchronous RESET Signal
- SER_IN: Serial bit stream received by unit
- ERROR: Output goes ‘1’ for Parity Error or Not getting Stop Bit after 16-bit
- statusOUT: Shows current status. For busy condition gives output ‘1’
- OUT16O: Output Bit vector in parallel form

VHDL Code

Architecture Block of UART_Receiver

```
architecture Behavioral of UART_Receiver is
signal status : std_logic := '0';
signal par : std_logic := '0';
signal OUT16 : std_logic_vector(15 downto 0):=(others=>'0');
signal CLK16x : std_logic;
signal count : integer := 0;
component CLK16
  port(CLK: in std_logic;
        enable: in std_logic:=‘0’;
       CLK16x: out std_logic);
end component;
begin
  CLKP : CLK16 port map(CLK, status, CLK16x);
  pmain : process(RESET, SER_IN, CLK16x, CLK)
  begin
    if RESET='1' then
      OUT16O <= x"0000";
      OUT16 <= x"0000";
      ERROR <= '0';
      count<=0;
      par<='0';
      status<='0';
      statusOUT <= '0';
```

→ Reset Block

VHDL Code

Start Bit to Parity bit Sampling

```
elsif CLK16x'event and CLK16x='1' then
  if status='0' and SER_IN='0' then
    status<='1';
    statusOUT <= '1';
    count <=0;
```

Getting Start Bit

```
  elsif status='1' and count<16 then
      count <= count+1;
      OUT16 <= OUT16(14 downto 0) & SER_IN;
```

Counting up and Shifting one bit to right until count reached to 16

```
      if SER_IN='1' then
        par <= not par;     --EVEN PARITY
      end if;
```

Update Parity if Input is '1'

```
    elsif count=16 and status='1' then
      count<=17;
      if SER_IN=(not par) then
            ERROR <= '1';
            OUT16 <= x"0000";
      end if;
```

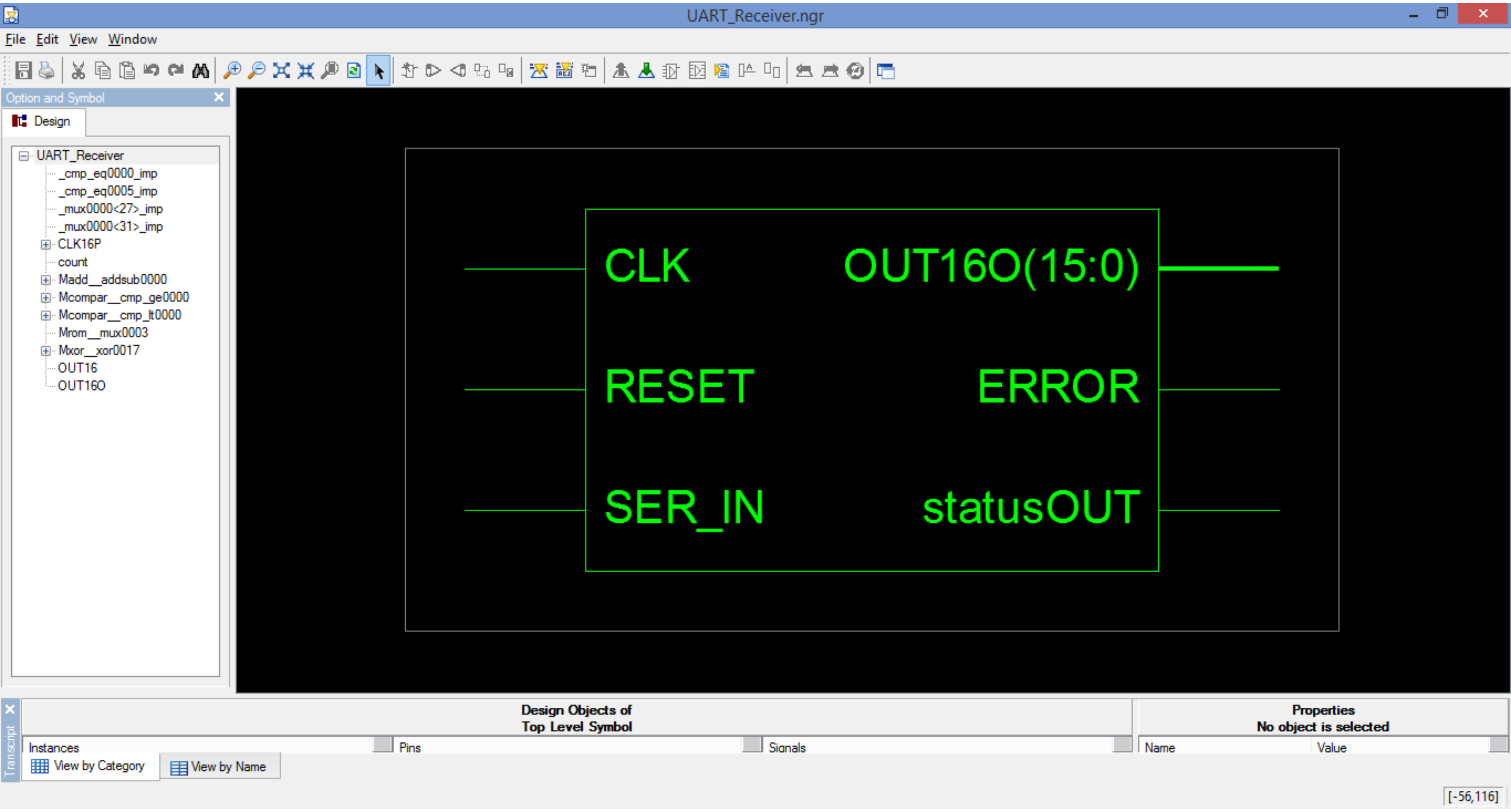
Check parity at last input i.e. count=16

VHDL Code

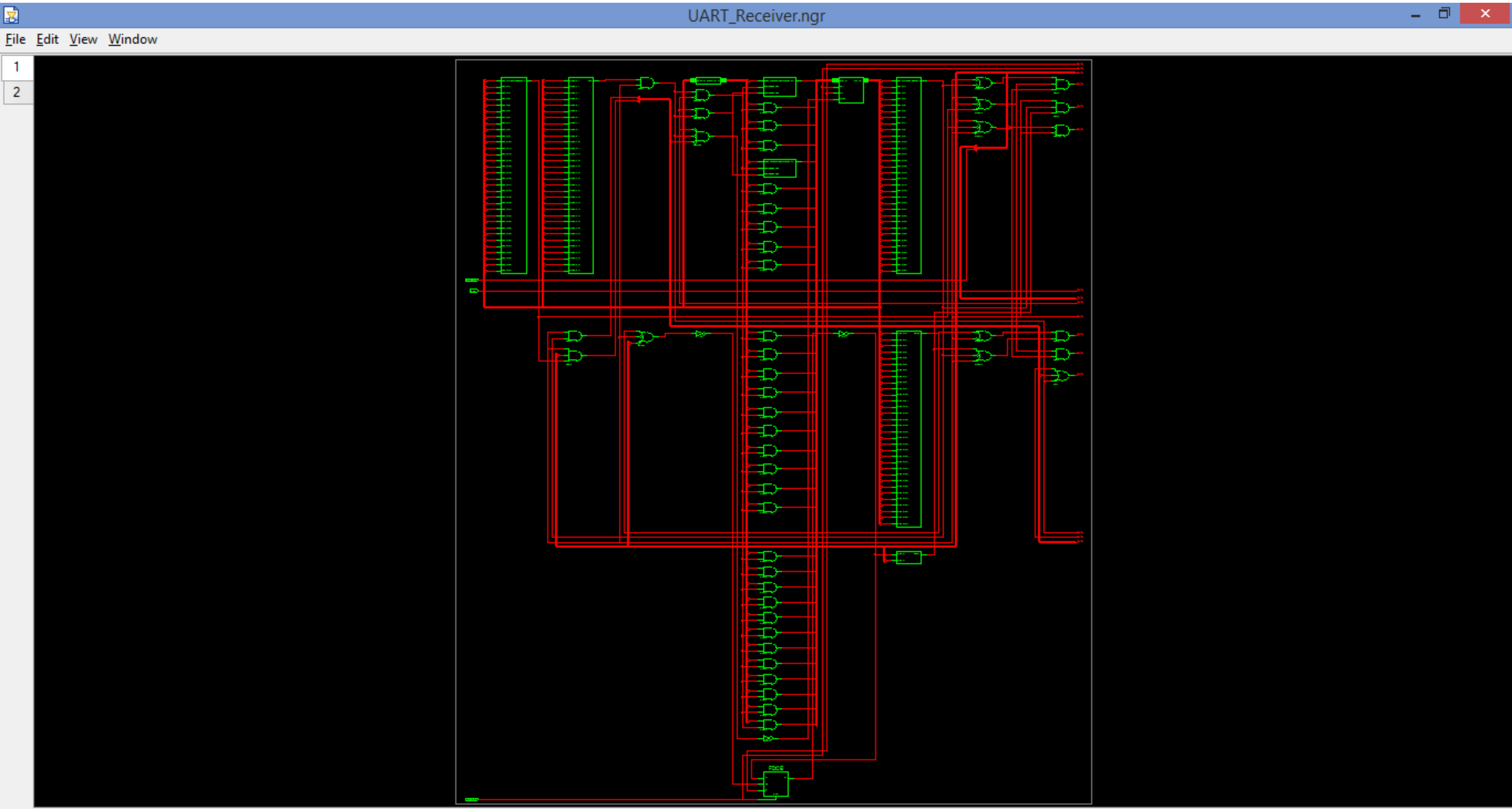
Sampling Stop bit and shift to IDLE state

```vhdl
elsif count=17 and SER_IN='1' and status='1' then
           --at 17th count, check for IDLE and reset
            OUT16O <= OUT16;
            ERROR <= '0';
            status<='0';
            statusOUT<='0';
        end if;
        elsif count=17 and SER_IN='0' and status='1' then
            OUT16O <= x"0000";
            ERROR <='1';
            status<='0';
            statusOUT<='0';
      end if;
   end process;
end Behavioral;
```

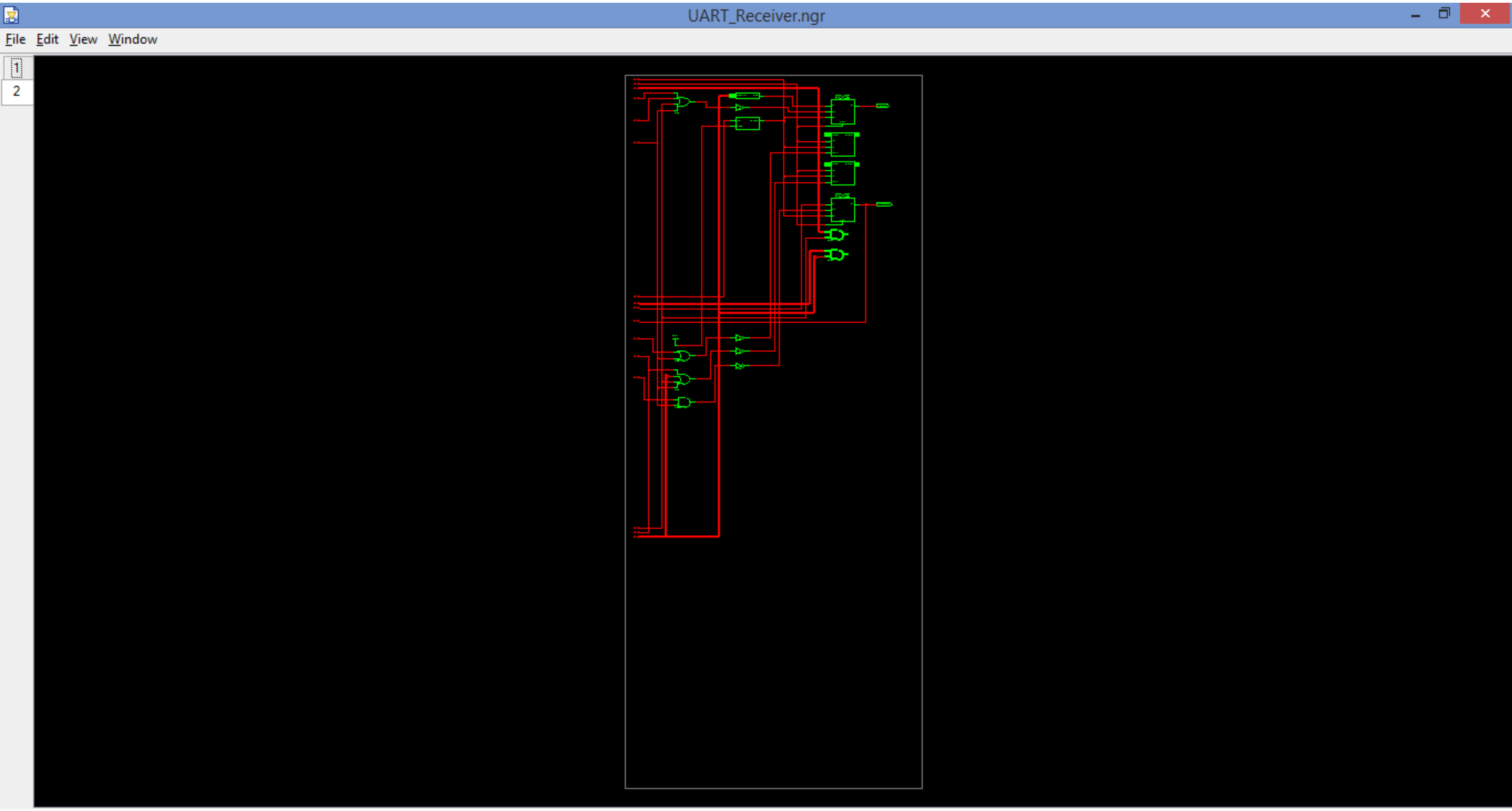
RTL Schematic : Block Diagram



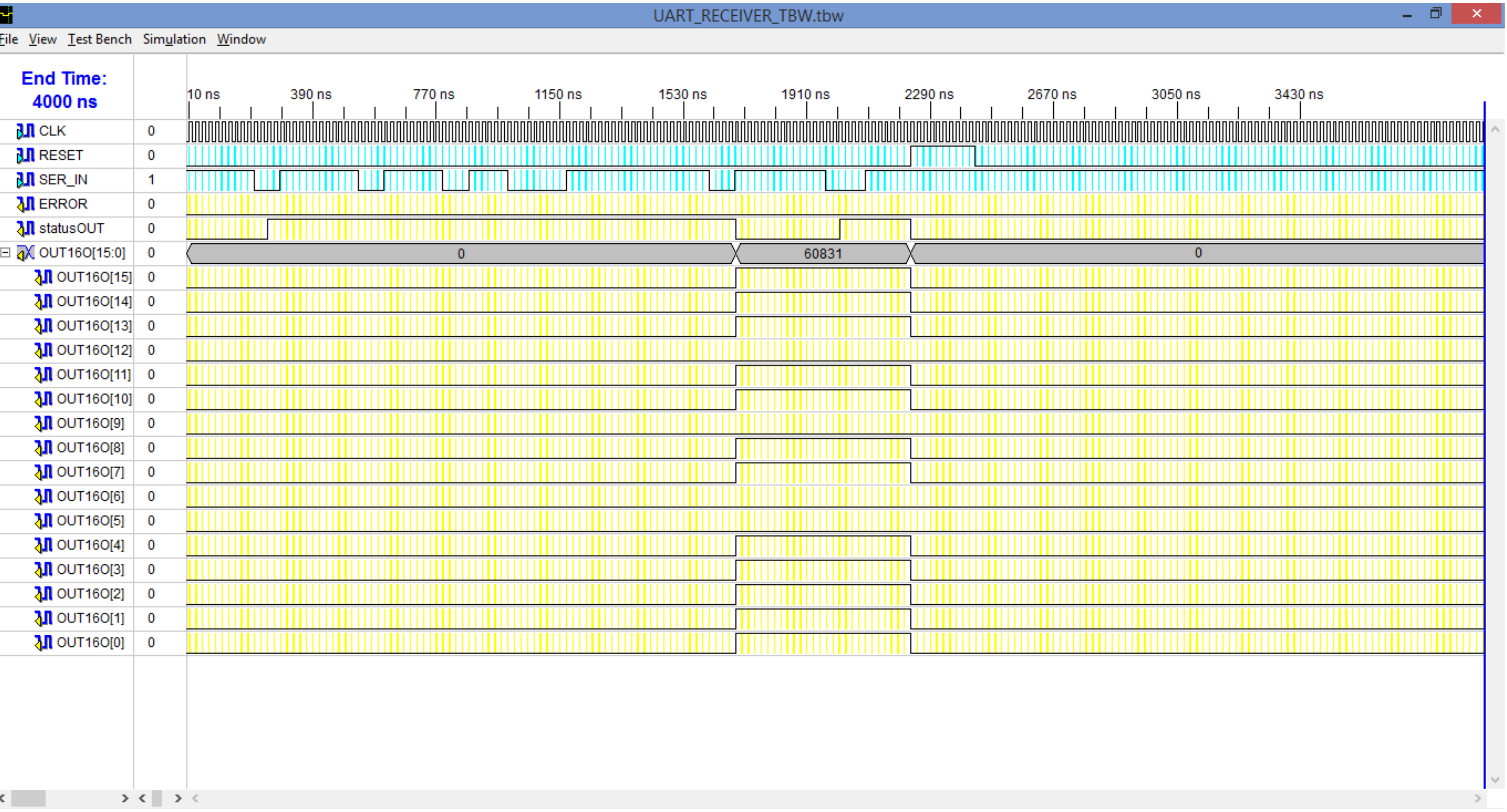
RTL Schematic : Inside Block Pg. 1



RTL Schematic : Inside Block Pg. 2



Simulation Results



For simulation Purpose, we have used 2 times baud rate as Clock

Summary

- UART Receiver is designed using VHDL and tested successfully on test bench simulation.
- Here clock cycle can be replaced by Baud Rate Pulses generated by Baud Rate Generator to accurately synchronise with Transmitter.

Thank you