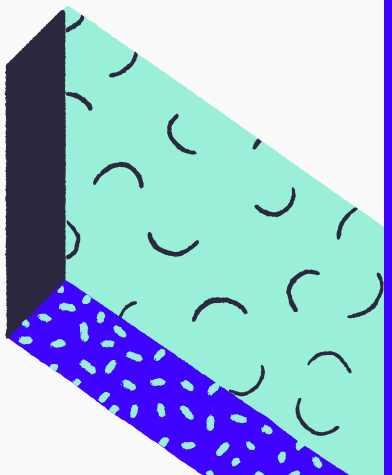
UART

IT'S STANDS FOR UNIVERSAL ASYNCHRONOUS RECEIVER/TRANSMITTER



UART is usually an individual (or part of an) integrated circuit (IC) used for serial communications over a computer or peripheral device serial port.

Hardware implementation of UART using SystemVerilog.

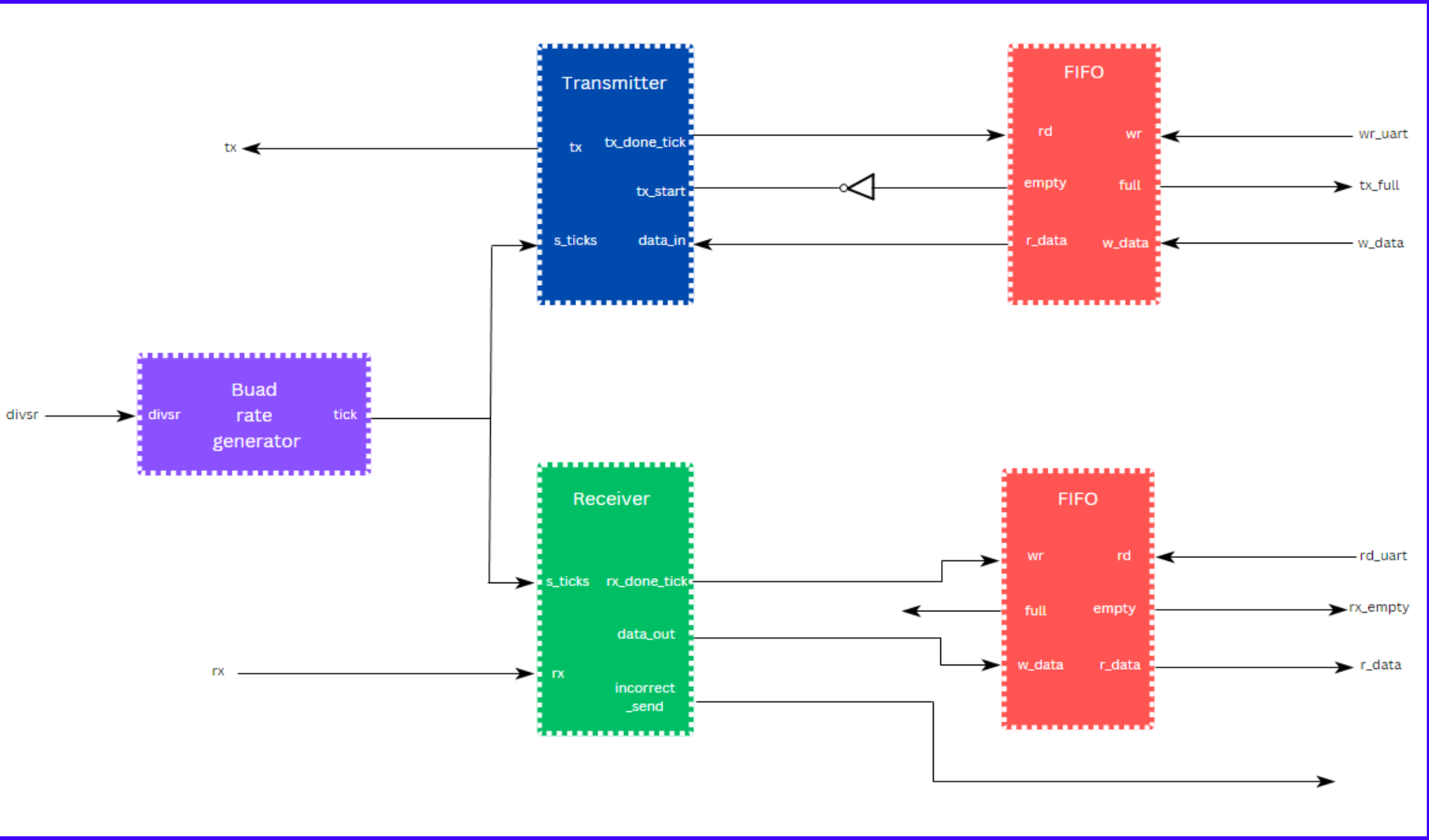
```
module UART #(

                                parameter Data_bits=9,
                          parameter Sp_ticks=16,
                          parameter St_ticks=8,
                          parameter Dt_ticks=16,
                                    parameter addr_width=5,
                                    parameter divsr_width=10,
                     parameter Read=2'b01,
                     parameter Write=2'b10,
                     parameter Read_and_Write=2'b11

               )

                                    (

                                    input clk,Reset,
                                    input rd_uart,wr_uart,
                                    input rx,
                                    input[Data_bits-2:0] w_data,
                                    input[divsr_width-1:0] divsr,
                                    output rx_empty,tx_full,
                                    output tx,
                                    output[Data_bits-2:0] r_data,
                                    output incorrect_send
);
```

```
module UART_TX #(
                parameter Data_bits=9, //including parity bit
                parameter Sp_ticks=16, //Stop_bit_ticks
                parameter St_ticks=8,  //Start_bit_ticks
                parameter Dt_ticks=16 //data ticks for transimting one data bit
      )

                (

       input clk,Reset,
                 input[Data_bits-2:0] data_in,
                 input tx_start,
                 input s_ticks,

                 output logic tx_done_tick,
                 output logic parity_check,
                 output logic tx

);

typedef enum  { idle , start , data , stop }  S_states;


S_states state_reg ,state_next; //to keep track of next state

logic tx_reg ,tx_next; //to keep track of transmitted data bit

logic[$clog2(Dt_ticks)-1:0] s_reg ,s_next; //to keep track of number of ticks

logic[$clog2(Data_bits)-1:0] n_reg ,n_next; //to keep track of number of transmitted bits

logic[Data_bits-2:0] sd_reg,sd_next; //data to be shifted
```

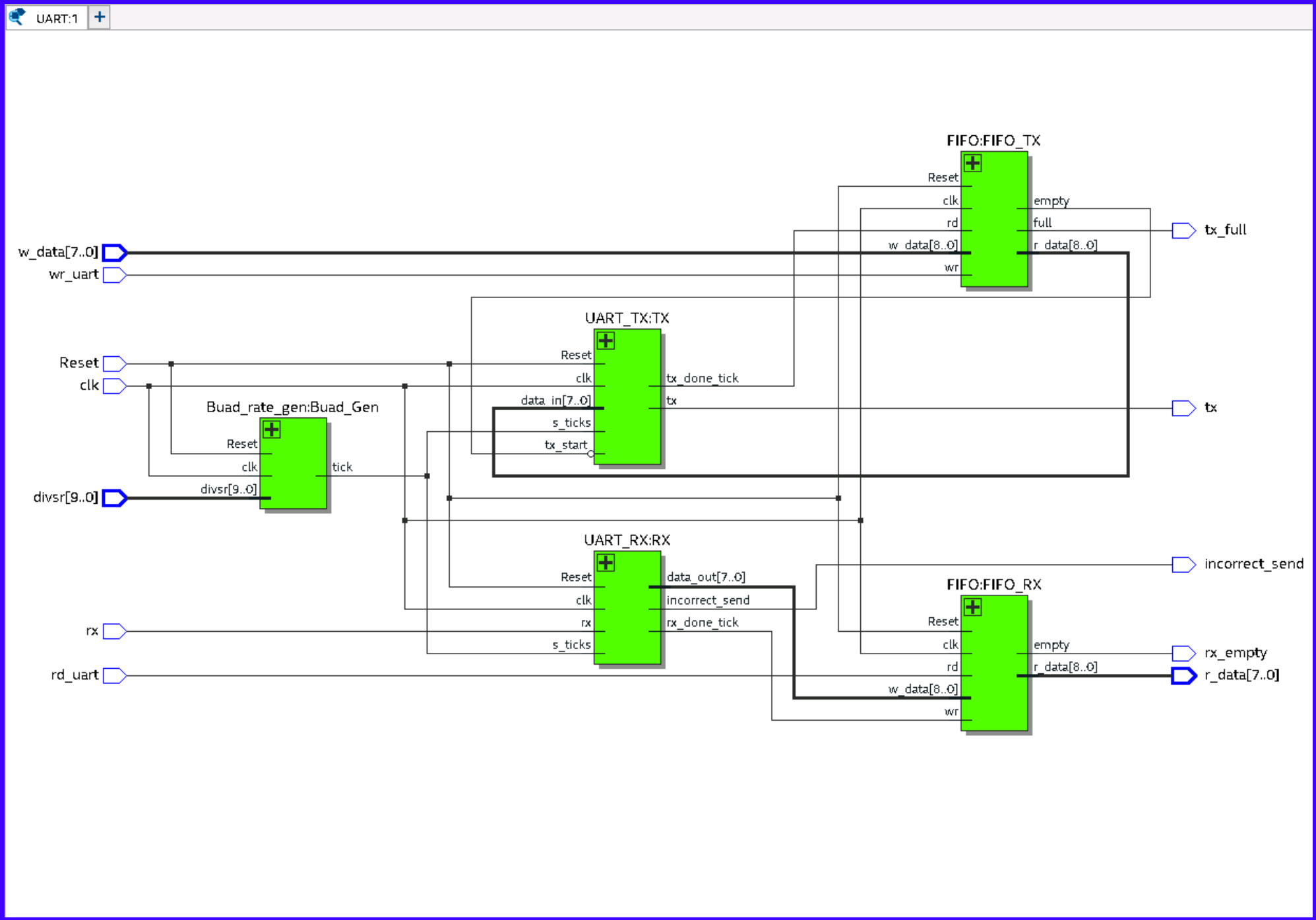
```
module UART_RX #( parameter Data_bits=9,
                  parameter Sp_ticks=16,//Stop_bit_ticks
                                        parameter St_ticks=8,//Start_bit_ticks
                                        parameter Dt_ticks=16//data ticks for received one data bit
       )

                 (
        input rx,
                  input clk,Reset,
                  input s_ticks,

                  output logic rx_done_tick,
                  output[Data_bits-2:0] data_out,
                  output logic incorrect_send // if it equals logic one the data is correct ,if it equal to zer


);


typedef enum {idle , start , data , stop } S_states;


S_states state_reg ,state_next;//to keep track of next state

logic[$clog2(Dt_ticks)-1:0] s_reg,s_next;//to keep track of number of ticks

logic[$clog2(Data_bits)-1:0] n_reg,n_next; //to keep track of number of received bits

logic[Data_bits-2:0] sd_reg,sd_next; //data to be shifted

logic parity_reg,parity_next;
```

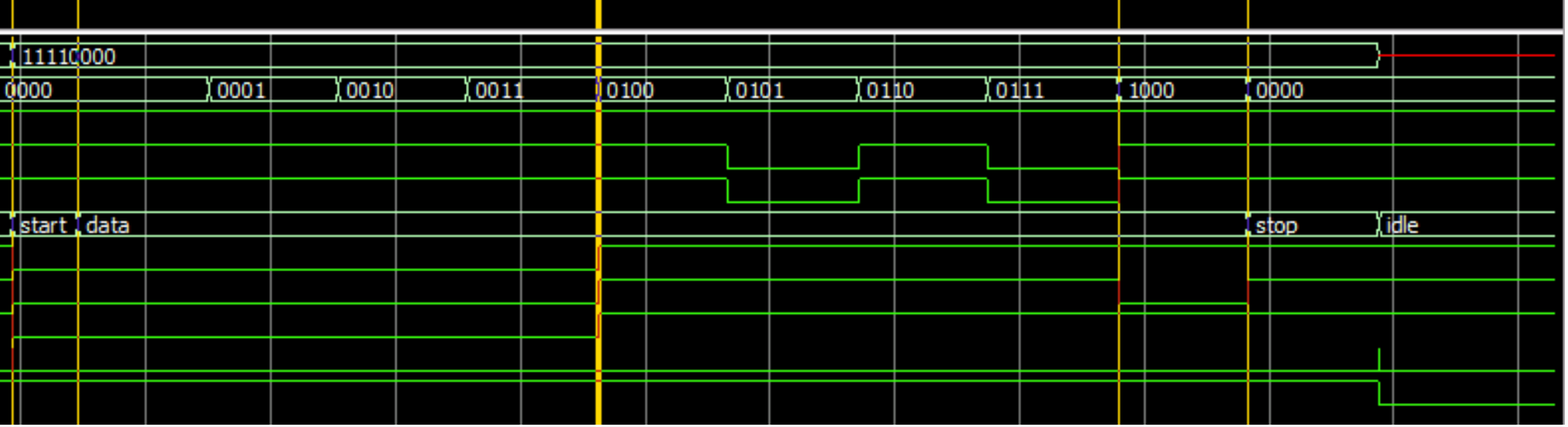
FOR MORE DETAILS CHECK MY GITHUB REPO

THE SCHEMATIC DESIGN

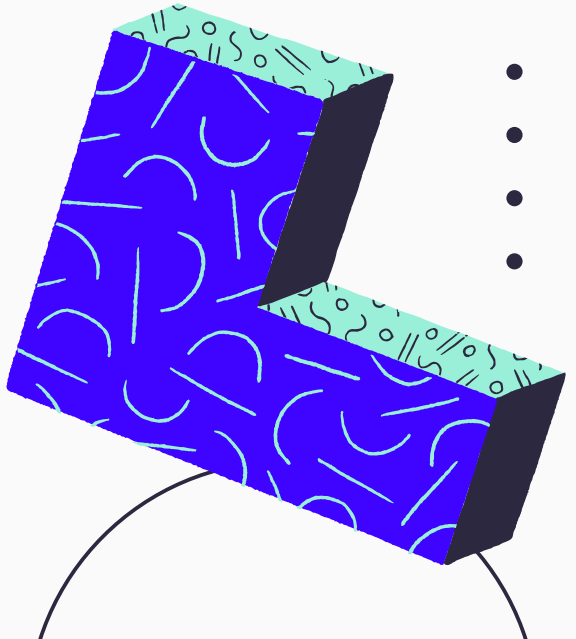


The Synthesis Result

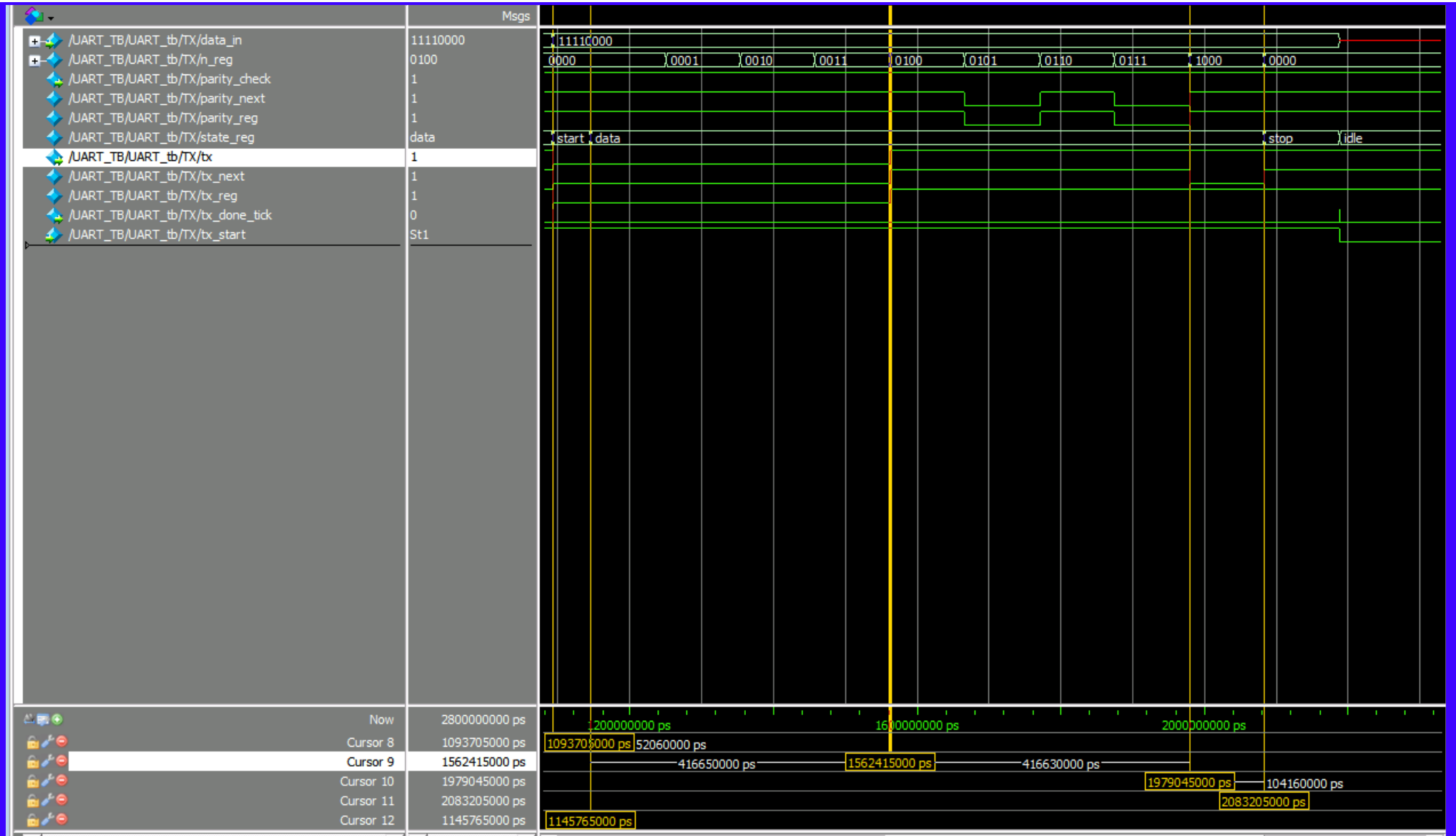

UART:1
FIFO:FIFO_TX
Reset
clk
rd
w_data[8..0]
wr
empty
full
r_data[8..0]
tx_full
w_data[7..0]
wr_uart
UART_TX:TX
Reset
clk
data_in[7..0]
s_ticks
tx_start
tx_done_tick
tx
Reset
clk
Buad_rate_gen:Buad_Gen
Reset
clk
divsr[9..0]
tick
divsr[9..0]
UART_RX:RX
Reset
clk
rx
s_ticks
data_out[7..0]
incorrect_send
rx_done_tick
incorrect_send
FIFO:FIFO_RX
Reset
clk
rd
w_data[8..0]
wr
empty
r_data[8..0]
rx_empty
r_data[7..0]
rx
rd_uart



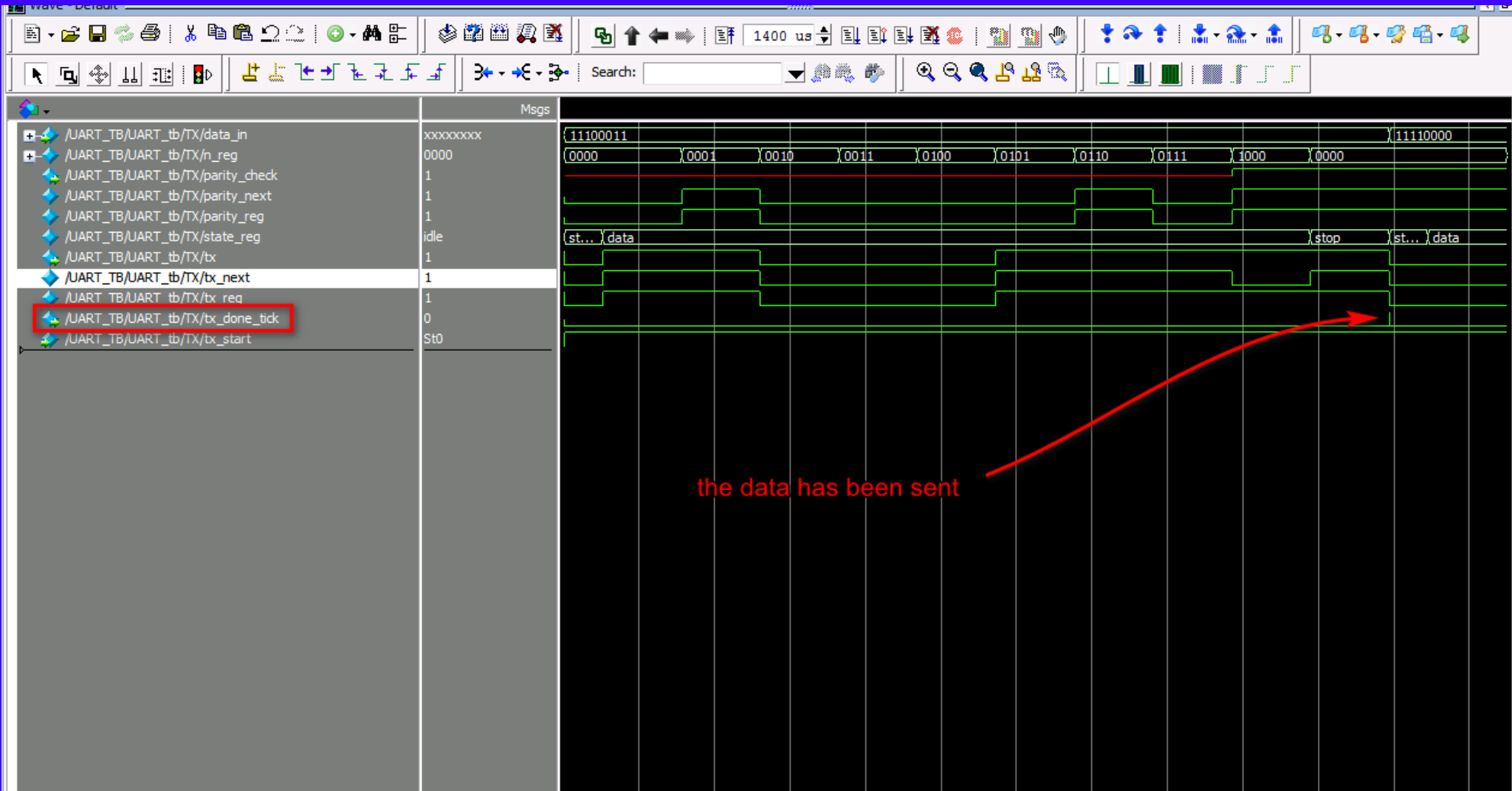
SIMULATION RESULT ON MODELSIM



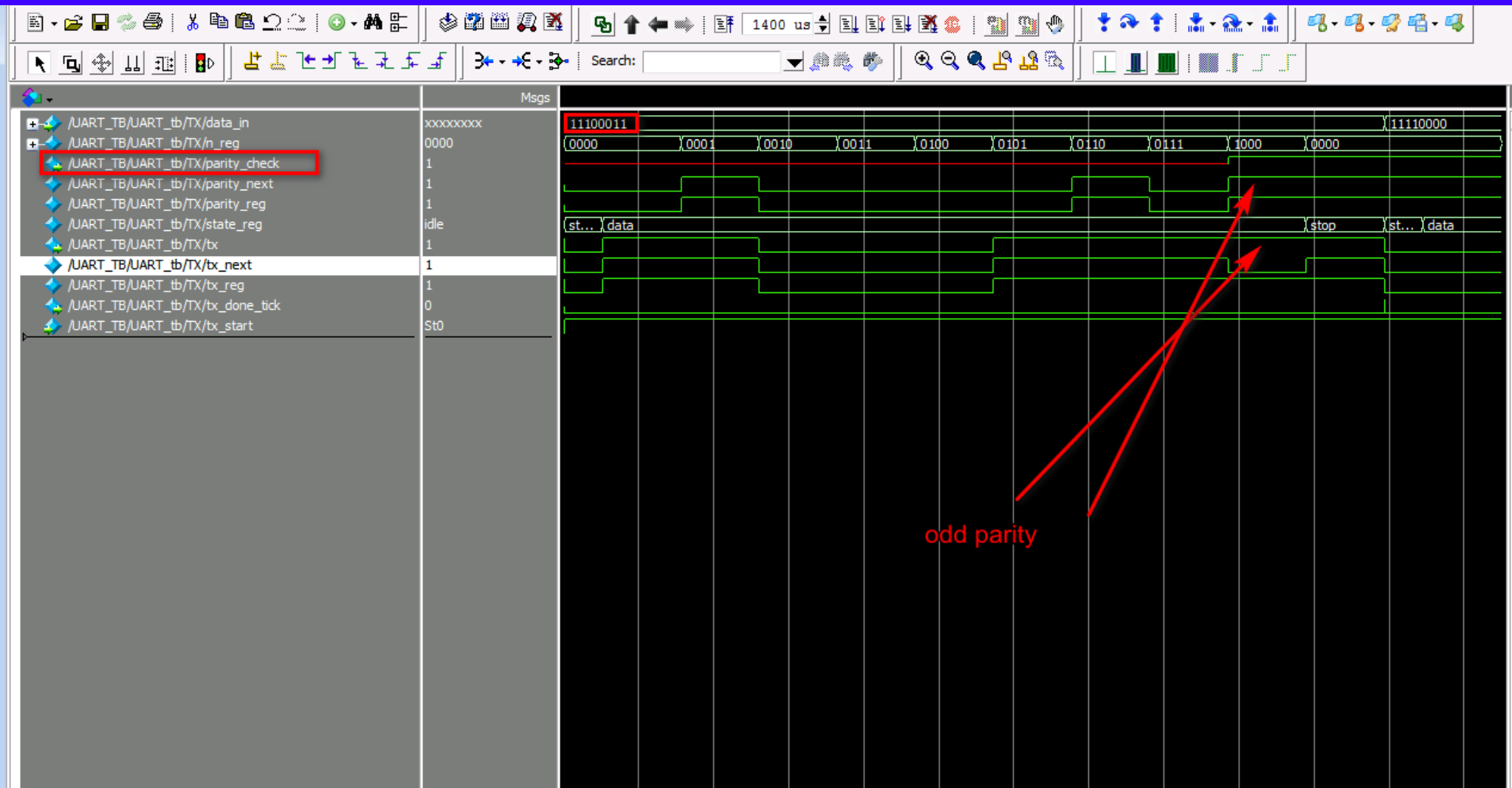
FIRST CASE


Msgs
/UART_TB/UART_tb/TX/data_in
/UART_TB/UART_tb/TX/n_reg
/UART_TB/UART_tb/TX/parity_check
/UART_TB/UART_tb/TX/parity_next
/UART_TB/UART_tb/TX/parity_reg
/UART_TB/UART_tb/TX/state_reg
/UART_TB/UART_tb/TX/tx
/UART_TB/UART_tb/TX/tx_next
/UART_TB/UART_tb/TX/tx_reg
/UART_TB/UART_tb/TX/tx_done_tick
/UART_TB/UART_tb/TX/tx_start
11110000
0100
1
1
1
data
1
1
1
0
St1
11110000
0000
0001
0010
0011
0100
0101
0110
0111
1000
0000
start
data
stop
idle
Now
2800000000 ps
Cursor 8
1093705000 ps
Cursor 9
1562415000 ps
Cursor 10
1979045000 ps
Cursor 11
2083205000 ps
Cursor 12
1145765000 ps
1200000000 ps
1600000000 ps
2000000000 ps
1093705000 ps
52060000 ps
416650000 ps
1562415000 ps
416630000 ps
1979045000 ps
104160000 ps
2083205000 ps
1145765000 ps

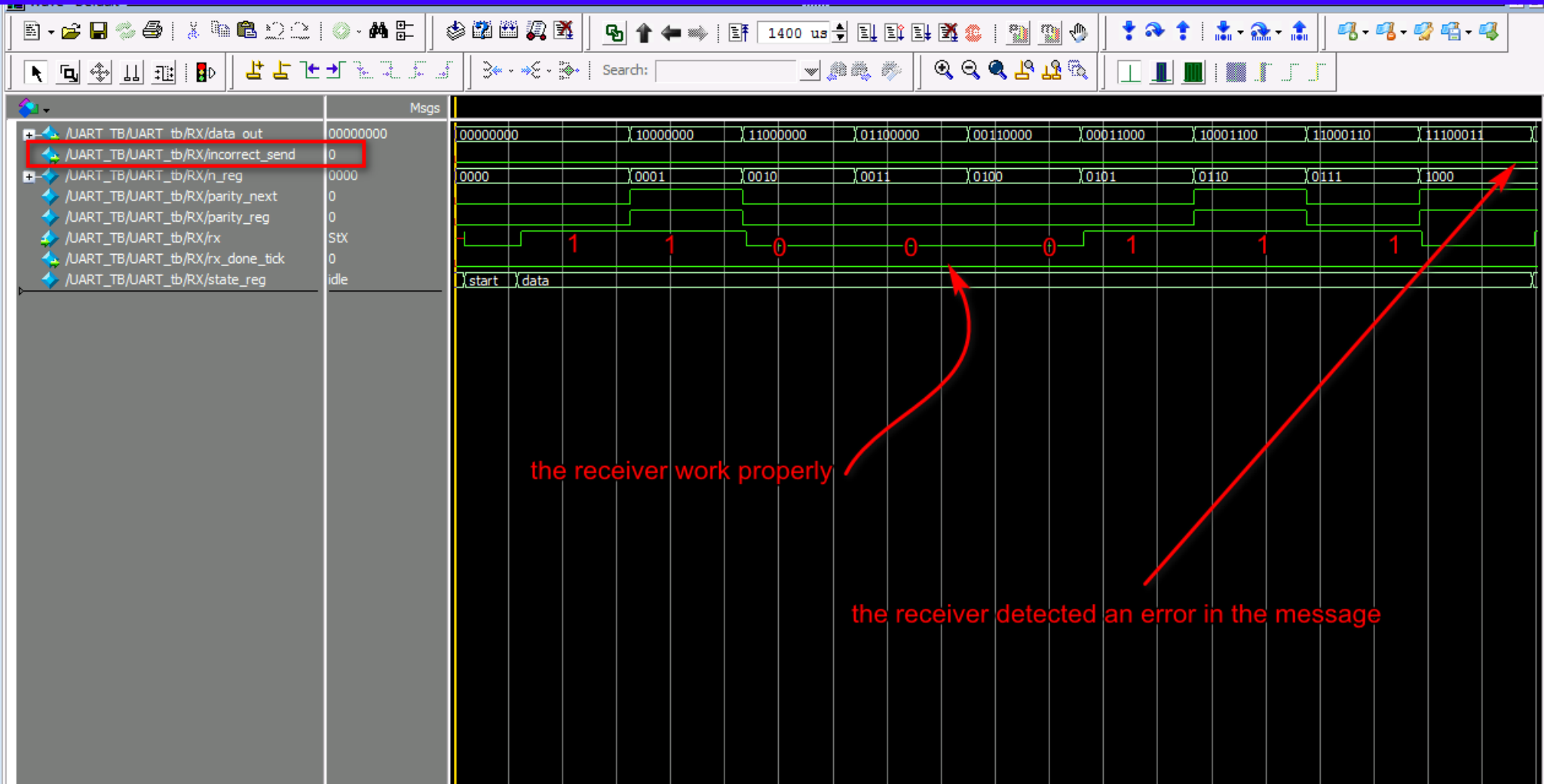
FIRST CASE


Msgs
/UART_TB/UART_tb/TX/data_in
/UART_TB/UART_tb/TX/n_reg
/UART_TB/UART_tb/TX/parity_check
/UART_TB/UART_tb/TX/parity_next
/UART_TB/UART_tb/TX/parity_reg
/UART_TB/UART_tb/TX/state_reg
/UART_TB/UART_tb/TX/tx
/UART_TB/UART_tb/TX/tx_next
/UART_TB/UART_tb/TX/tx_reg
/UART_TB/UART_tb/TX/tx_done_tick
/UART_TB/UART_tb/TX/tx_start
xxxxxxxx
0000
1
1
1
idle
1
1
1
0
St0
11100011
11110000
0000
0001
0010
0011
0100
0101
0110
0111
1000
0000
st...
data
stop
st...
data
the data has been sent

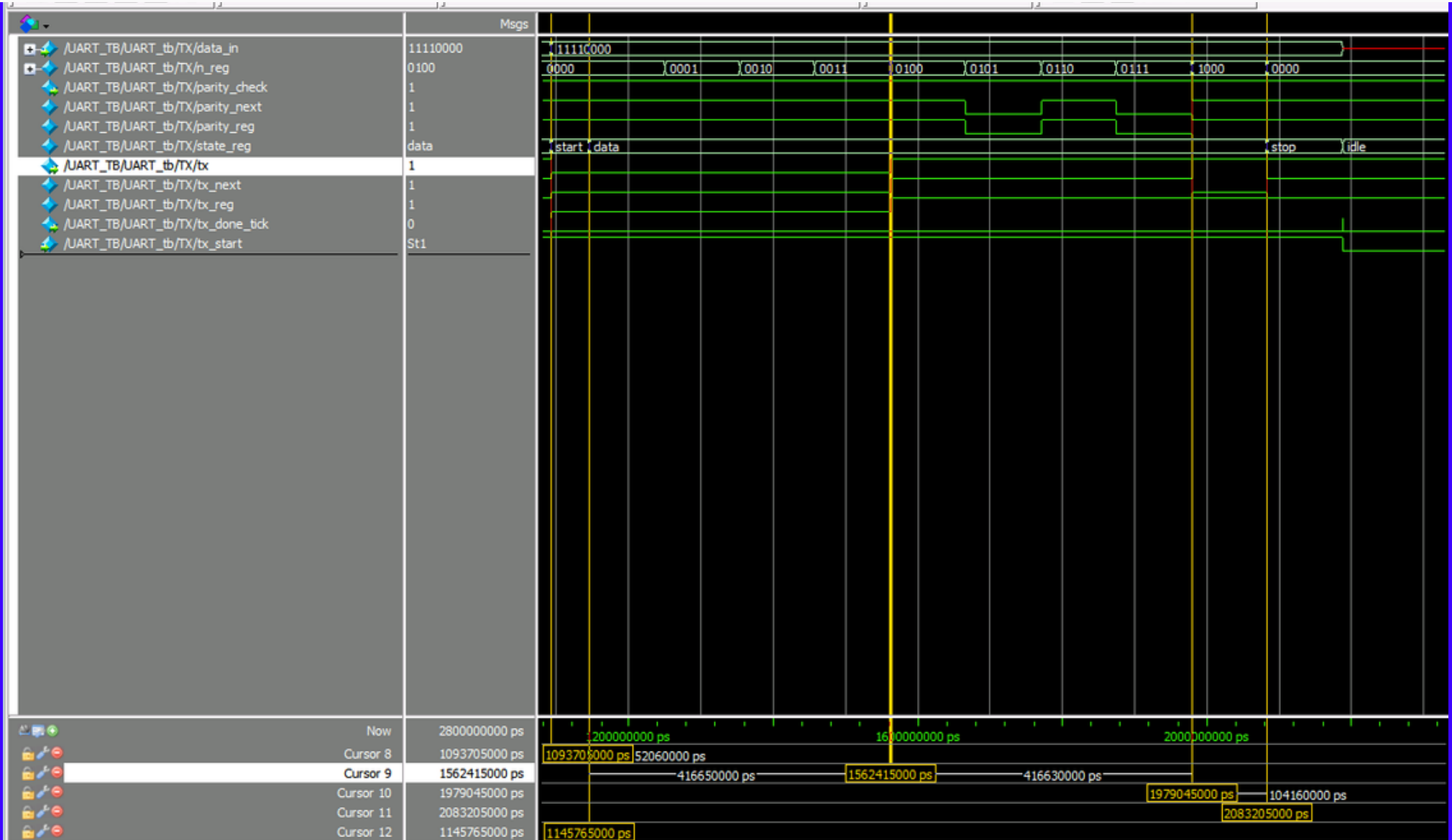
FIRST CASE


Msgs
/UART_TB/UART_tb/TX/data_in
/UART_TB/UART_tb/TX/n_reg
/UART_TB/UART_tb/TX/parity_check
/UART_TB/UART_tb/TX/parity_next
/UART_TB/UART_tb/TX/parity_reg
/UART_TB/UART_tb/TX/state_reg
/UART_TB/UART_tb/TX/tx
/UART_TB/UART_tb/TX/tx_next
/UART_TB/UART_tb/TX/tx_reg
/UART_TB/UART_tb/TX/tx_done_tick
/UART_TB/UART_tb/TX/tx_start
xxxxxxxx
0000
1
1
1
idle
1
1
1
0
St0
11100011
11110000
0000
0001
0010
0011
0100
0101
0110
0111
1000
0000
st...
data
stop
st...
data
odd parity

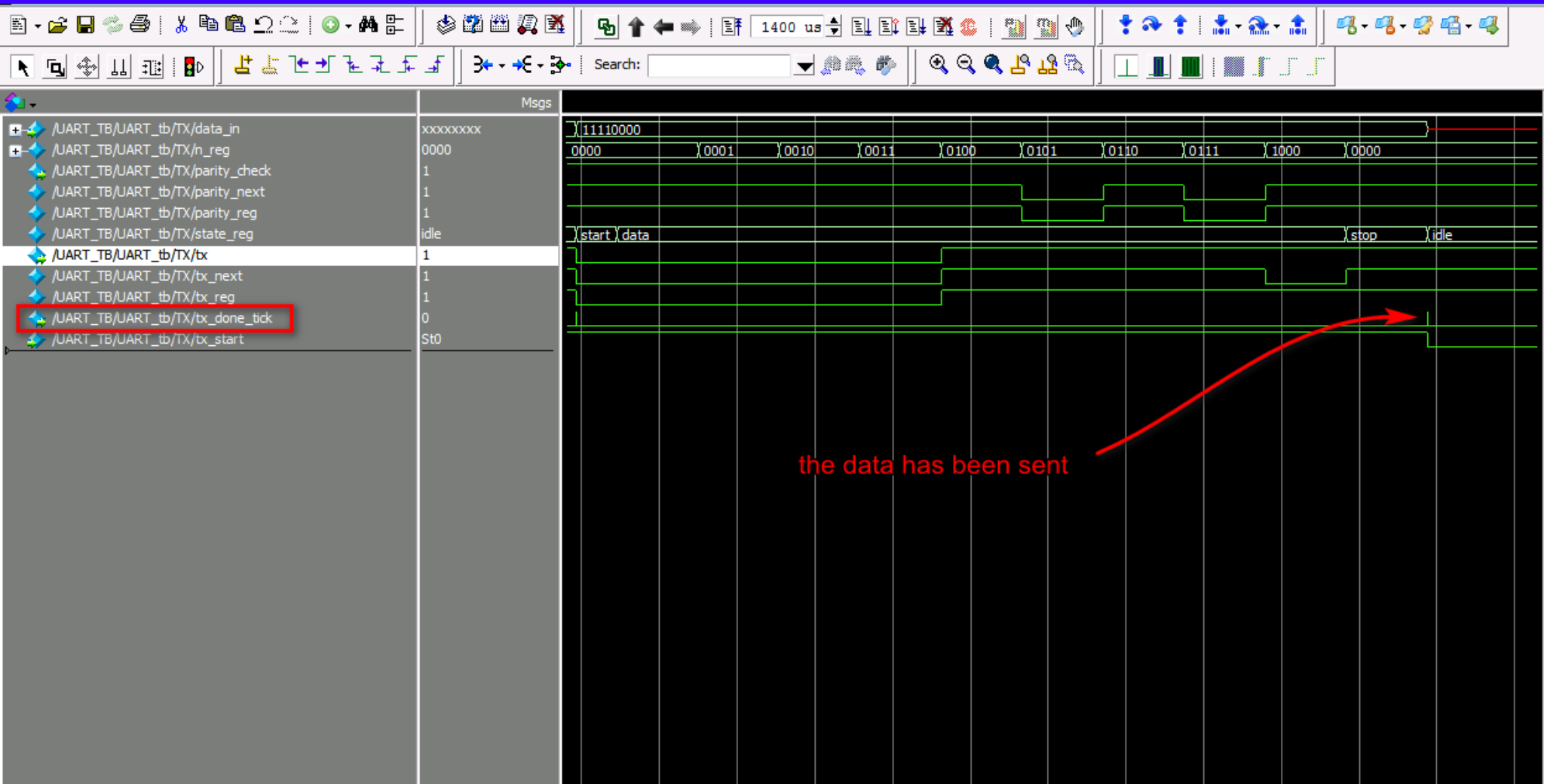
FIRST CASE


1400 us
Search:
Msgs
/UART_TB/UART_tb/RX/data_out
/UART_TB/UART_tb/RX/incorrect_send
/UART_TB/UART_tb/RX/n_reg
/UART_TB/UART_tb/RX/parity_next
/UART_TB/UART_tb/RX/parity_reg
/UART_TB/UART_tb/RX/rx
/UART_TB/UART_tb/RX/rx_done_tick
/UART_TB/UART_tb/RX/state_reg
00000000
0
0000
0
0
StX
0
idle
00000000
10000000
11000000
01100000
00110000
00011000
10001100
11000110
11100011
0000
0001
0010
0011
0100
0101
0110
0111
1000
1
1
0
0
0
1
1
1
start
data
the receiver work properly
the receiver detected an error in the message

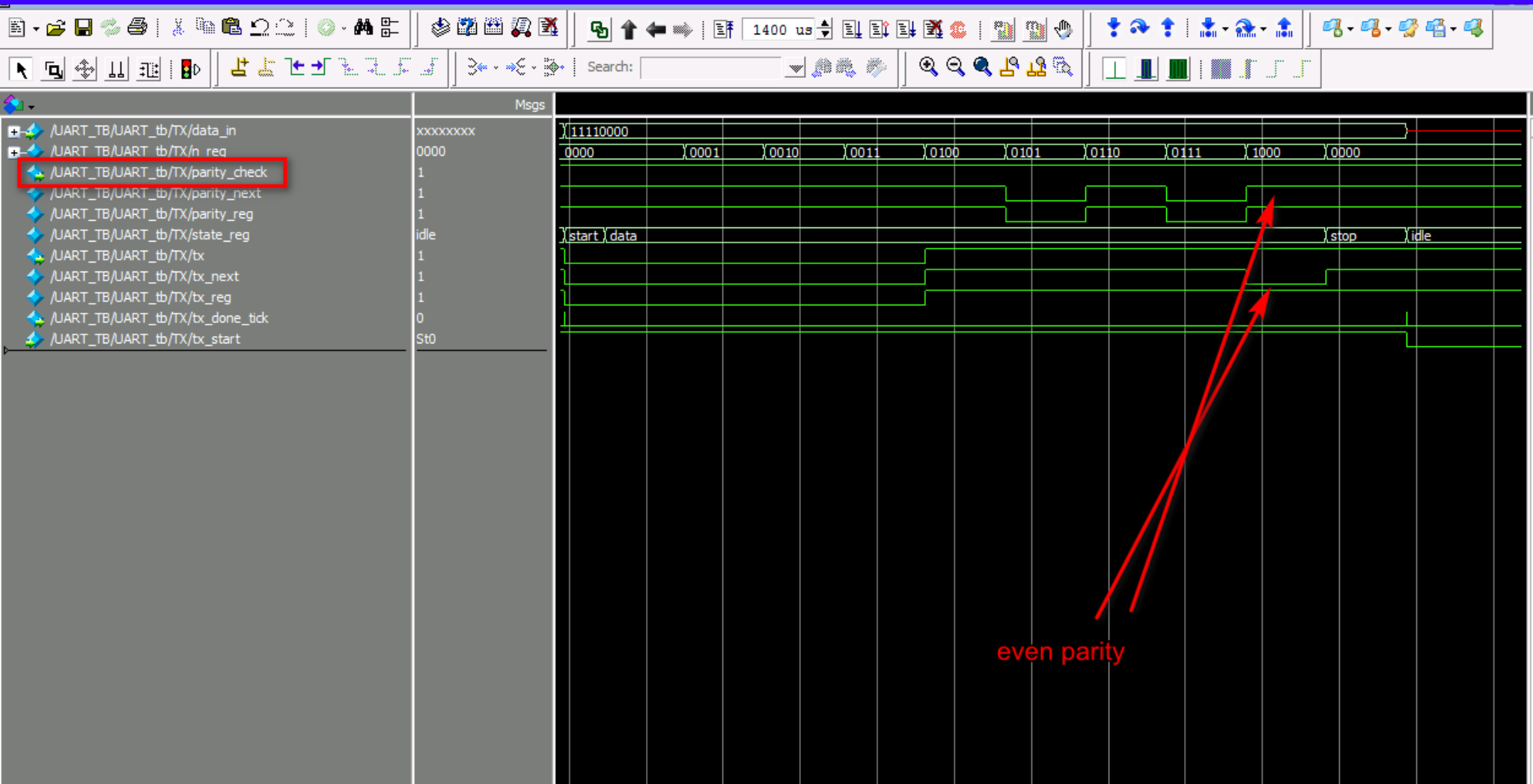
SECOND CASE


Msgs
/UART_TB/UART_tb/TX/data_in
/UART_TB/UART_tb/TX/n_reg
/UART_TB/UART_tb/TX/parity_check
/UART_TB/UART_tb/TX/parity_next
/UART_TB/UART_tb/TX/parity_reg
/UART_TB/UART_tb/TX/state_reg
/UART_TB/UART_tb/TX/tx
/UART_TB/UART_tb/TX/tx_next
/UART_TB/UART_tb/TX/tx_reg
/UART_TB/UART_tb/TX/tx_done_tick
/UART_TB/UART_tb/TX/tx_start
11110000
0100
1
1
1
data
1
1
1
0
St1
11110000
0000
0001
0010
0011
0100
0101
0110
0111
1000
0000
start
data
stop
idle
Now
2800000000 ps
Cursor 8
1093705000 ps
Cursor 9
1562415000 ps
Cursor 10
1979045000 ps
Cursor 11
2083205000 ps
Cursor 12
1145765000 ps
52060000 ps
416650000 ps
416630000 ps
104160000 ps

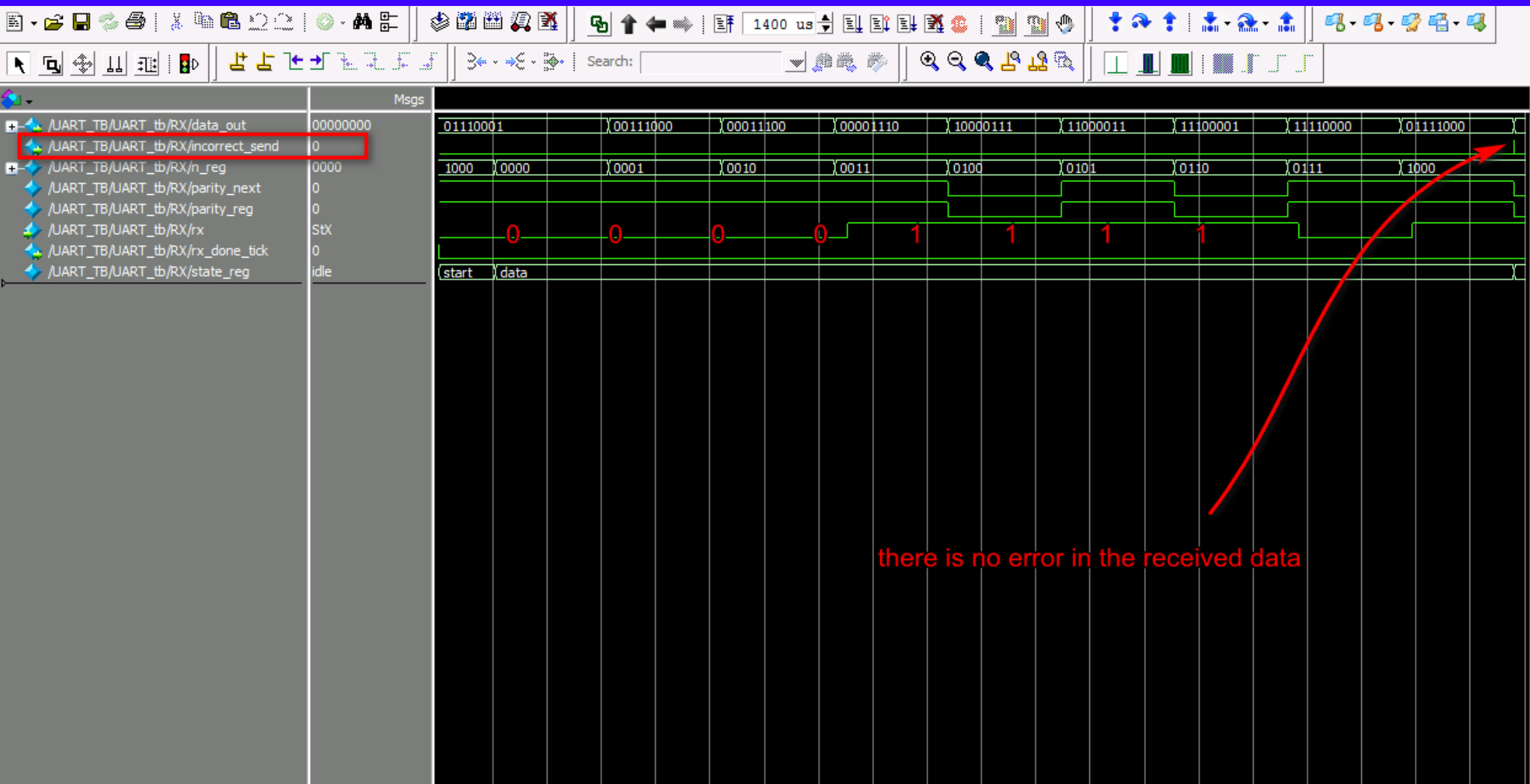
SECOND CASE


Msgs
/UART_TB/UART_tb/TX/data_in
/UART_TB/UART_tb/TX/n_reg
/UART_TB/UART_tb/TX/parity_check
/UART_TB/UART_tb/TX/parity_next
/UART_TB/UART_tb/TX/parity_reg
/UART_TB/UART_tb/TX/state_reg
/UART_TB/UART_tb/TX/tx
/UART_TB/UART_tb/TX/tx_next
/UART_TB/UART_tb/TX/tx_reg
/UART_TB/UART_tb/TX/tx_done_tick
/UART_TB/UART_tb/TX/tx_start
xxxxxxxx
0000
1
1
1
idle
1
1
1
0
St0
11110000
0000
0001
0010
0011
0100
0101
0110
0111
1000
0000
start
data
stop
idle
the data has been sent

SECOND CASE


Msgs
/UART_TB/UART_tb/TX/data_in
/UART_TB/UART_tb/TX/n_reg
/UART_TB/UART_tb/TX/parity_check
/UART_TB/UART_tb/TX/parity_next
/UART_TB/UART_tb/TX/parity_reg
/UART_TB/UART_tb/TX/state_reg
/UART_TB/UART_tb/TX/tx
/UART_TB/UART_tb/TX/tx_next
/UART_TB/UART_tb/TX/tx_reg
/UART_TB/UART_tb/TX/tx_done_tick
/UART_TB/UART_tb/TX/tx_start
xxxxxxxx
0000
1
1
1
idle
1
1
1
0
St0
11110000
0000
0001
0010
0011
0100
0101
0110
0111
1000
0000
start
data
stop
idle
even parity

SECOND CASE


Msgs
/UART_TB/UART_tb/RX/data_out
/UART_TB/UART_tb/RX/incorrect_send
/UART_TB/UART_tb/RX/n_reg
/UART_TB/UART_tb/RX/parity_next
/UART_TB/UART_tb/RX/parity_reg
/UART_TB/UART_tb/RX/rx
/UART_TB/UART_tb/RX/rx_done_tick
/UART_TB/UART_tb/RX/state_reg
00000000
0
0000
0
0
StX
0
idle
01110001
00111000
00011100
00001110
10000111
11000011
11100001
11110000
01111000
1000
0000
0001
0010
0011
0100
0101
0110
0111
1000
start
data
there is no error in the received data

THANK YOU FOR WATCHING!