# UART RX

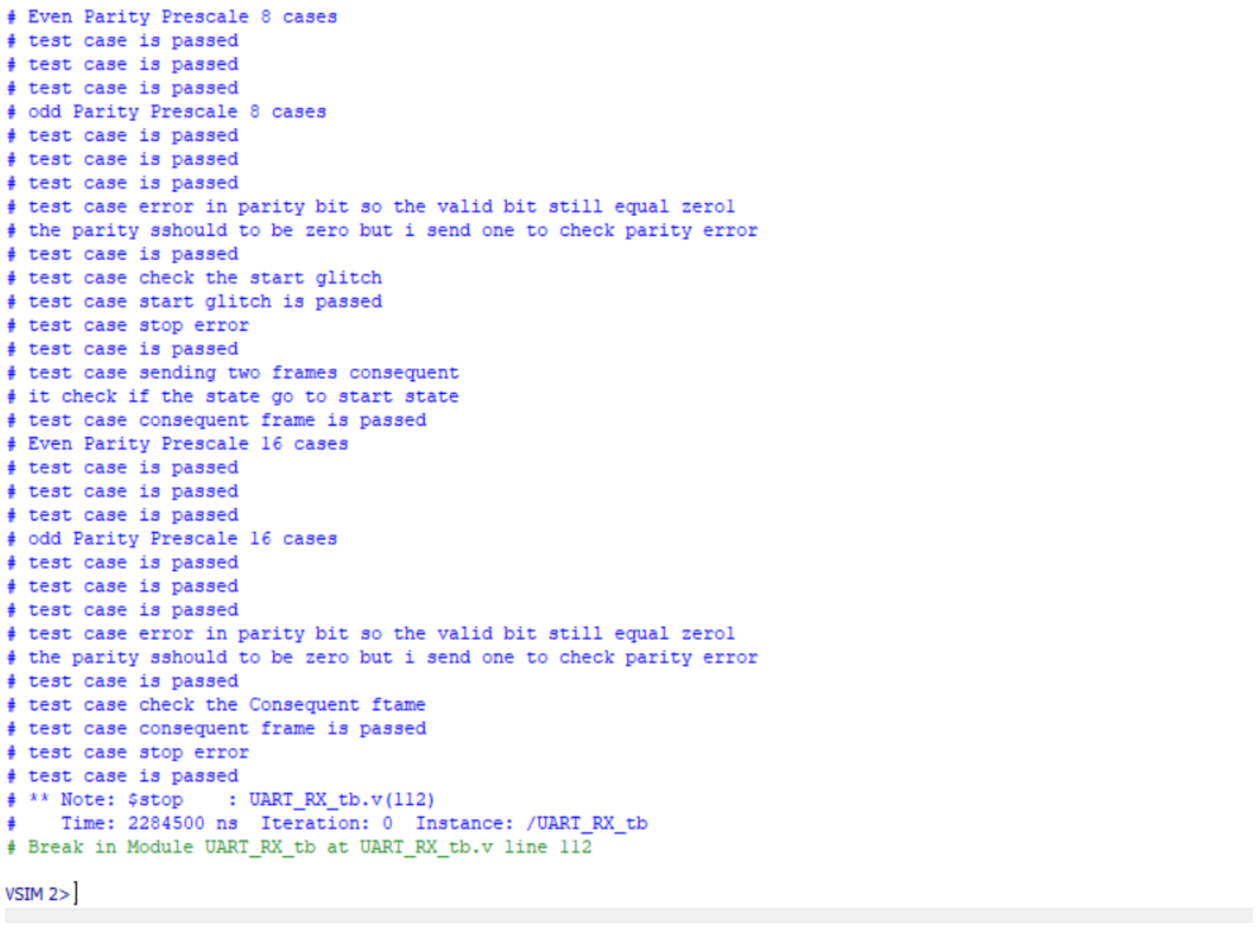
Eng:ali el-temsah
Eng:Hassan Khaled

## Introduction

Universal Asynchronous Receiver/Transmitter (UART) is a widely used communication protocol in embedded systems, facilitating serial communication between devices. The UART RX (Receiver) component is responsible for receiving serial data, converting it into parallel data, and making it available for further processing. Testing the UART RX is crucial to ensure reliable data communication, and a testbench is employed for this purpose.

A testbench is a virtual environment used to simulate and verify the functionality of a digital design. For a UART RX, the testbench simulates the transmission of serial data to the receiver and checks if the data is correctly received and interpreted.

## TEST CASES



```
# Even Parity Prescale 8 cases
# test case is passed
# test case is passed
# test case is passed
# odd Parity Prescale 8 cases
# test case is passed
# test case is passed
# test case is passed
# test case error in parity bit so the valid bit still equal zerol
# the parity sshould to be zero but i send one to check parity error
# test case is passed
# test case check the start glitch
# test case start glitch is passed
# test case stop error
# test case is passed
# test case sending two frames consequent
# it check if the state go to start state
# test case consequent frame is passed
# Even Parity Prescale 16 cases
# test case is passed
# test case is passed
# test case is passed
# odd Parity Prescale 16 cases
# test case is passed
# test case is passed
# test case is passed
# test case error in parity bit so the valid bit still equal zerol
# the parity sshould to be zero but i send one to check parity error
# test case is passed
# test case check the Consequent ftame
# test case consequent frame is passed
# test case stop error
# test case is passed
# ** Note: $stop    : UART_RX_tb.v(112)
#    Time: 2284500 ns  Iteration: 0  Instance: /UART_RX_tb
# Break in Module UART_RX_tb at UART_RX_tb.v line 112

VSIM 2>
```

*Figure 1 all test cases.*

## Test cases of prescale 8

### Test cases Even parity

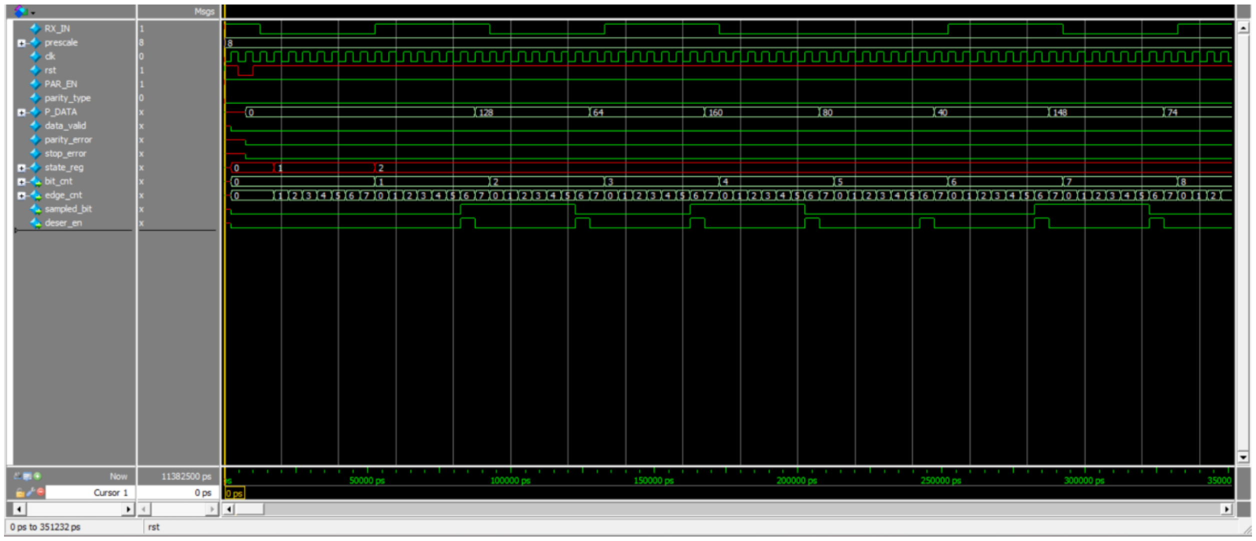


*Figure 2 illustrate reset and how data sampling.*

As shown in figure firstly the reset signal is triggered to low then transmitted data is be low to synchronized with the receiver as start bit the UART RX is works faster than the transmitter to make oversampling to ensure data received correctly, in the wave form the edge counter count from zero to seven then bit count is incremented by one and the sampled bit calculated after the sampling block finished then the deserializer enable is on on data sampling state and so on .

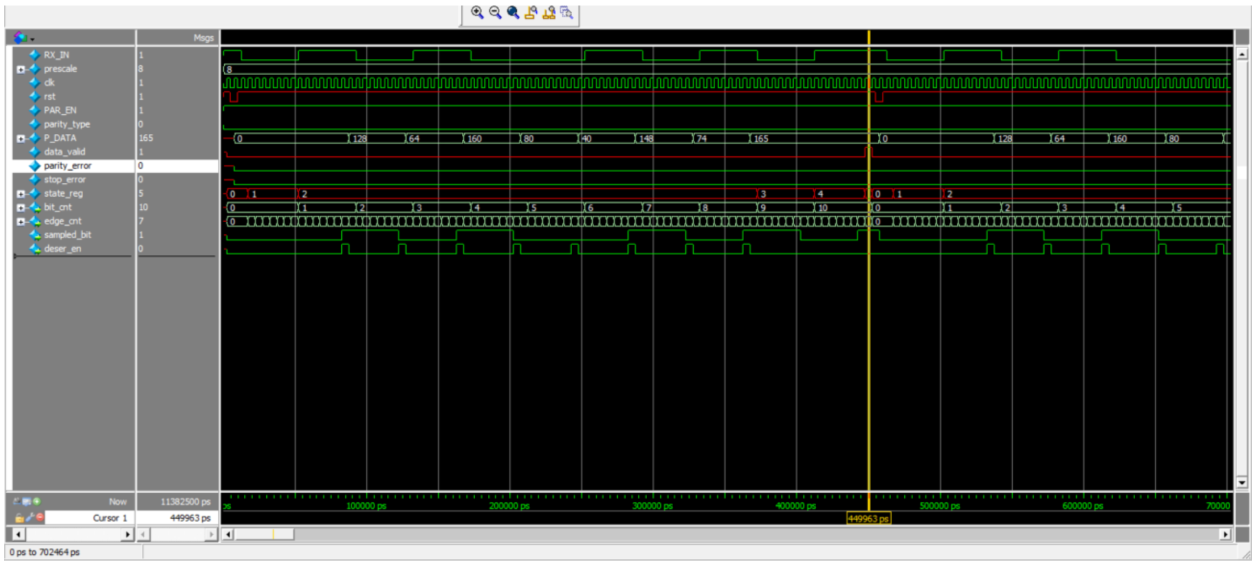


*Figure 3 complete frame.*

As shown in figure after sampling 8 bits (data) we go into state check parity and calculated parity is equal to transmitted parity bit so error parity stay equal zero then we go into state check stop and as shown the stop bit sampled equal to one so error stop stay equal to zero , because of there is no error in parity and stop the valid bit is triggered to be one and the frame is finished so he goes again to idle state and so on .

Note: I have made state with name consequent frame state at the last edge counter to check valid bit will be logic one or zero and to check the receiver RX date is zero to go the start state without go to idle state

## Another frame with correct data

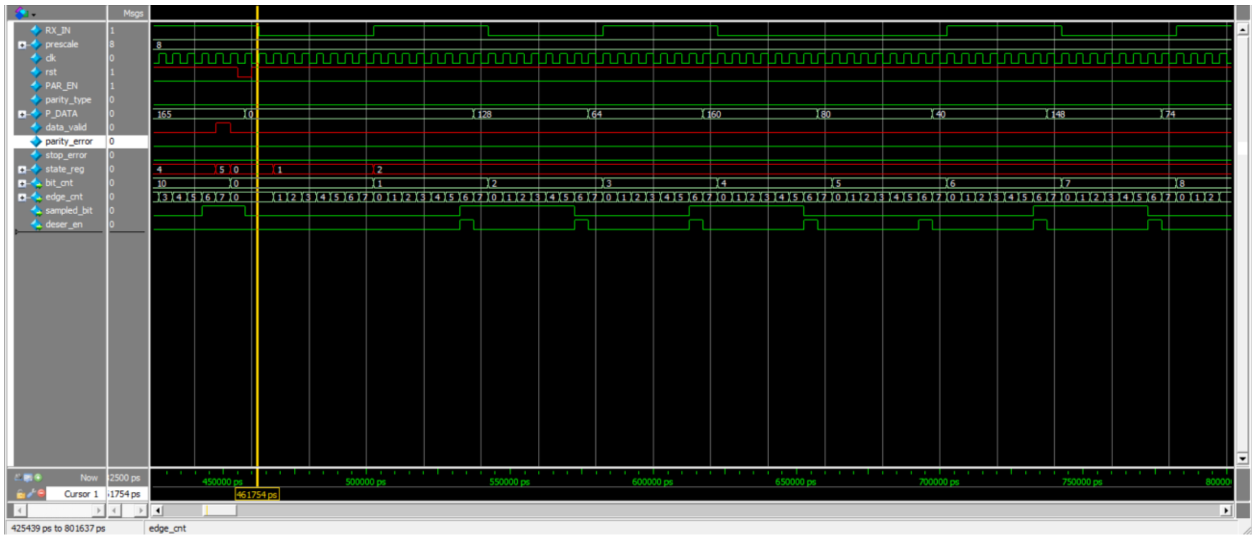


*Figure 4 another frame with correct data.*

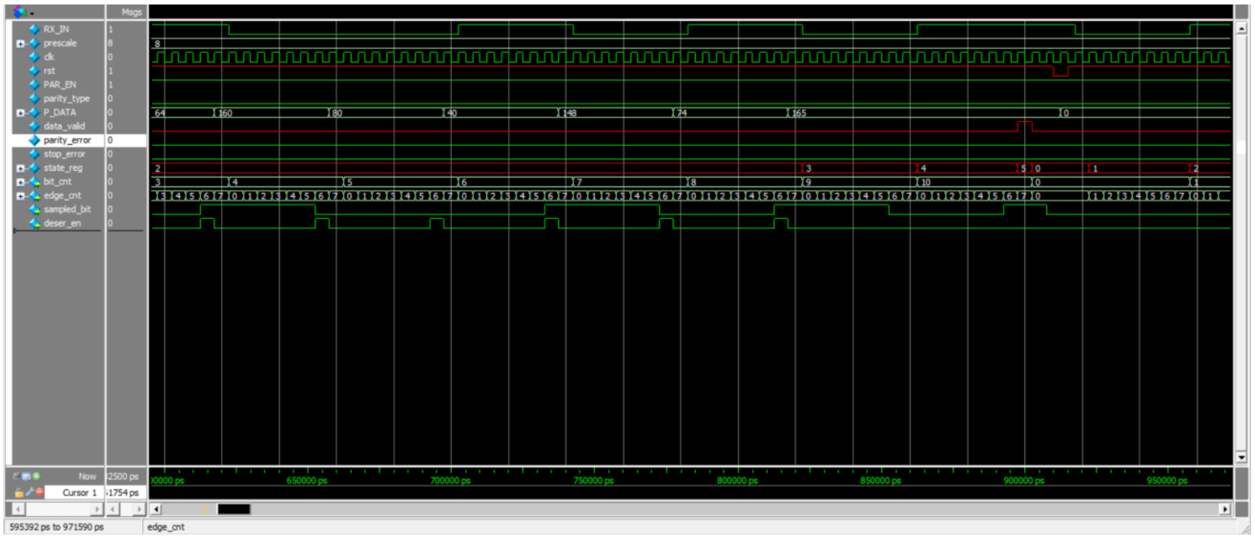


*Figure 5 the valid trigger of the frame.*

## Odd parity test cases

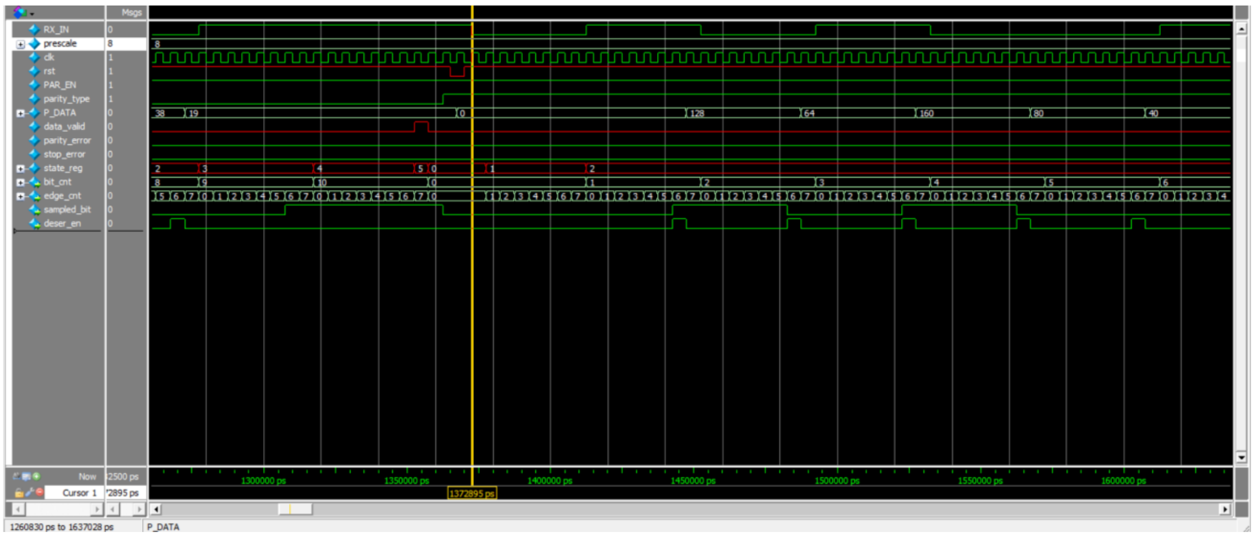


*Figure 6 changing parity type.*

Here we trigger the parity bit to be one (odd type).

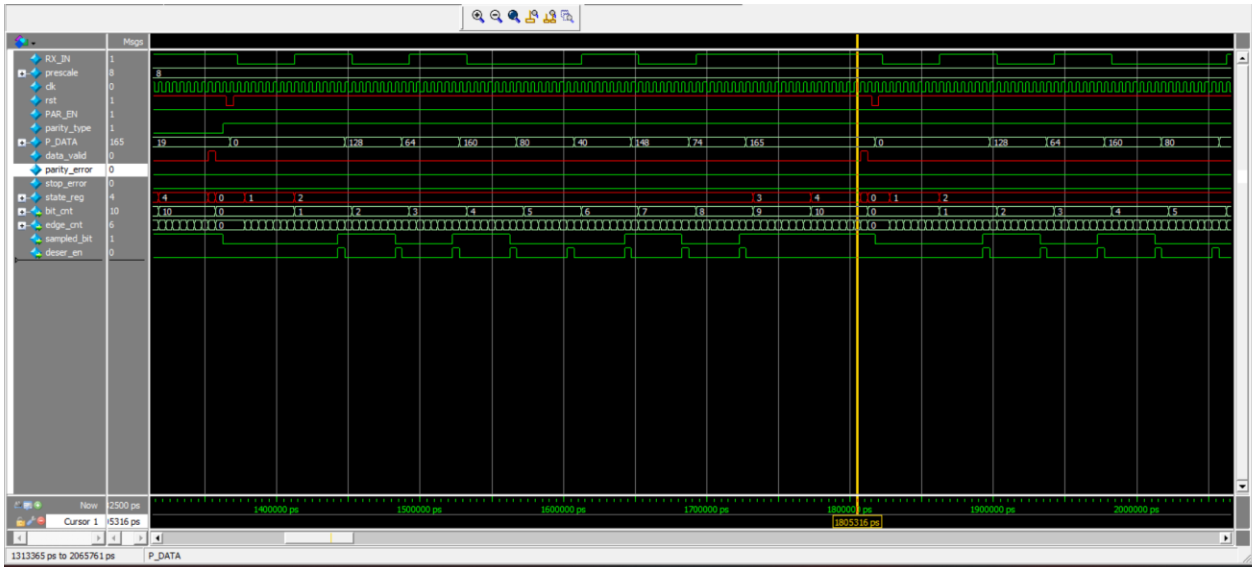


*Figure 7 complete frame of check odd parity.*

After calculating the parity bit of the data frame using XNOR it equals the sampled bit of transmitted parity bit so valid bit is triggered to logic one.

## Test case error in parity

The send frame is 11'b01010111011 and the type of parity is odd so this frame should have parity bit equal zero, but I send one to check parity error flag

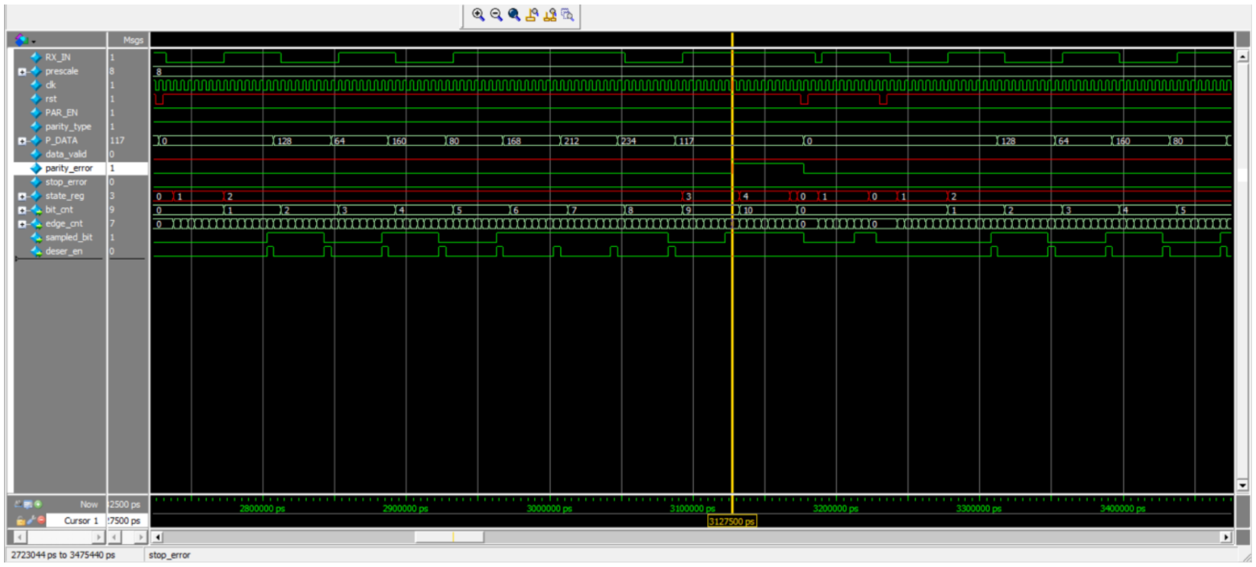


*Figure 8 parity error frame.*

As shown in the figure the parity error flag is changed to one and the valid bit did not change it still equal zero, so this frame is not valid.

## Test case start glitch

I will change the RX data to logic zero for one clock cycle only as a glitch

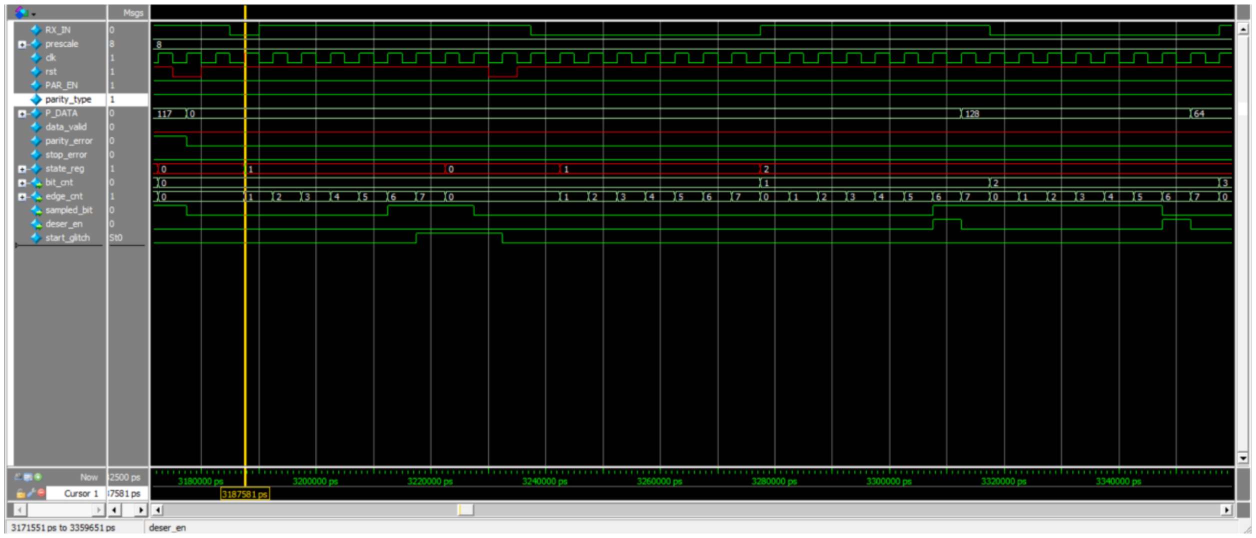


*Figure 9 start glitch check.*

After the sampling block finished and he found this glitch and the sampled bit is equal to one not zero, so he works correctly and start glitch flag is raised and state go to idle state again not data sampling state.

## Test case Stop check

I have sent frame 11'b01010101010 the stop bit I have sent it equal zero not one the stop error flag should be raised

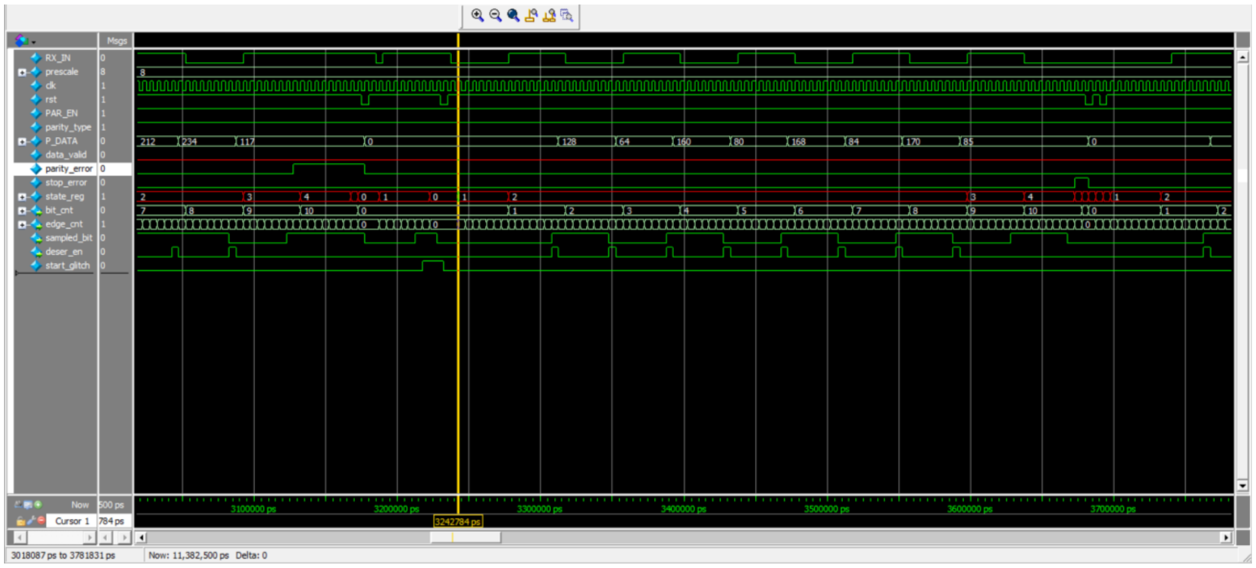


*Figure 10 stop error state.*

The stop error flag is raised, and the valid bit is still zero.

## The consequent frame test case

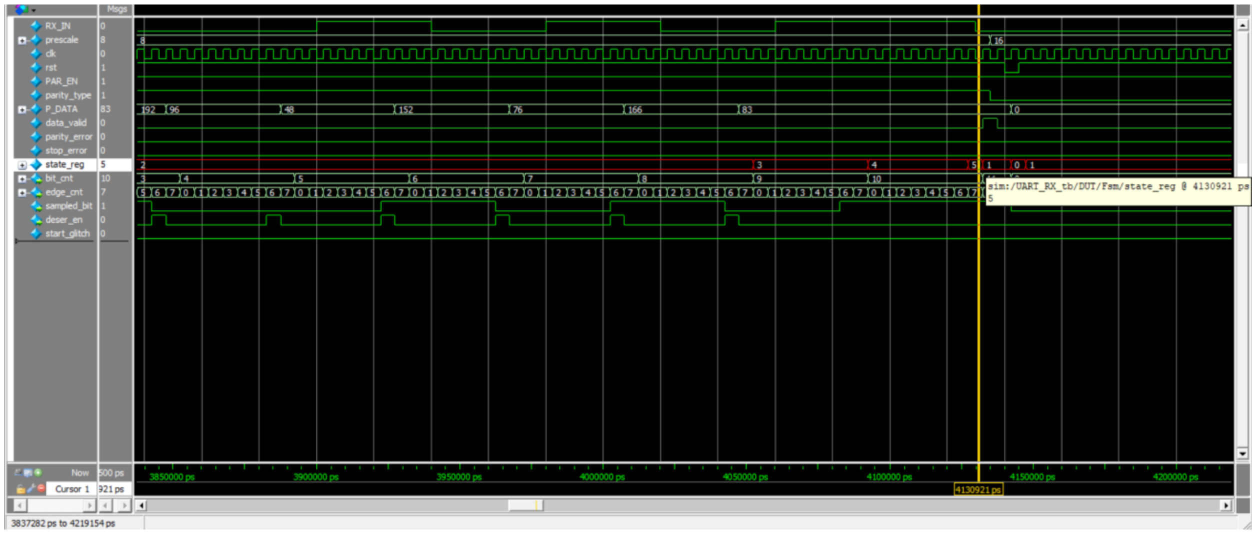


*Figure 11 consequent frames check.*

I have sent zero after the stop bit, so I go to start state not idle state.

## Test case Prescale 16

I have repeated the same cases but with prescale 16 , so the number of clock cycles for sampling is 16 clock cycles and so on.

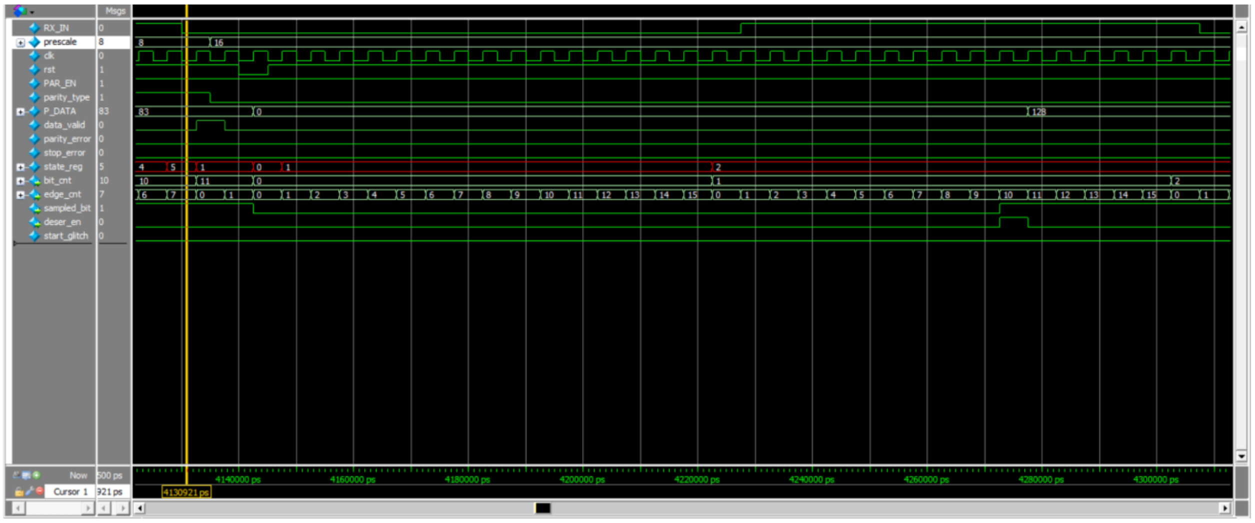


*Figure 12prescale 16.*

Same flow but edge counter is count from 0 to 16 the bit counter is incremented.

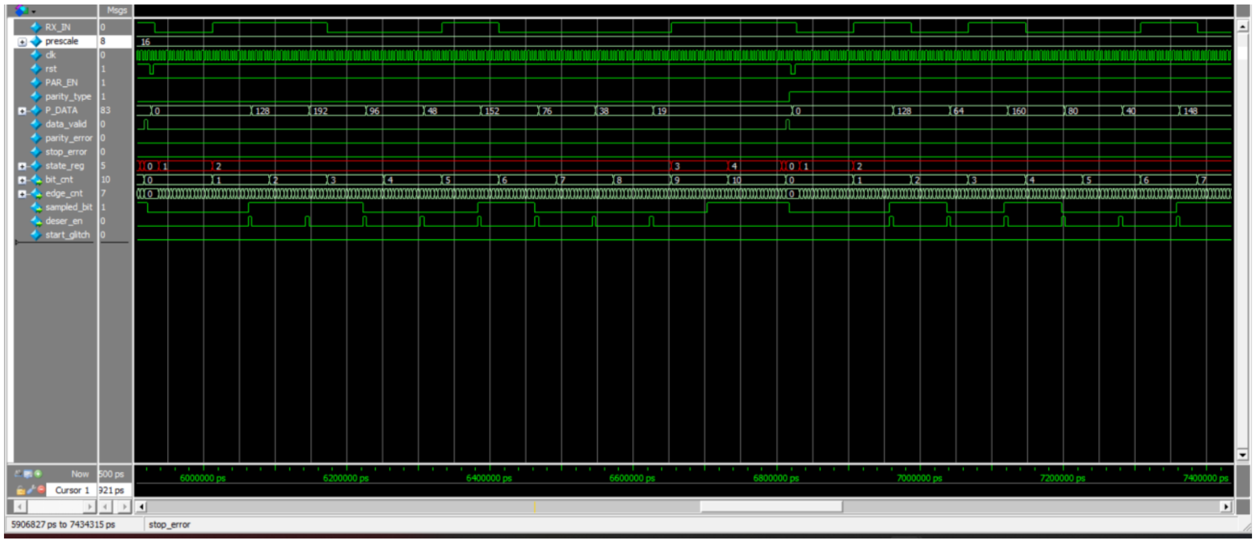


*Figure 13 complete frame with prescaler 16.*