Assignment 05 – UART receiver

EDB HDL

ANM, v03

Introduction

UART is a well known digital interface. In this assignment a UART receiver unit shall be developed.

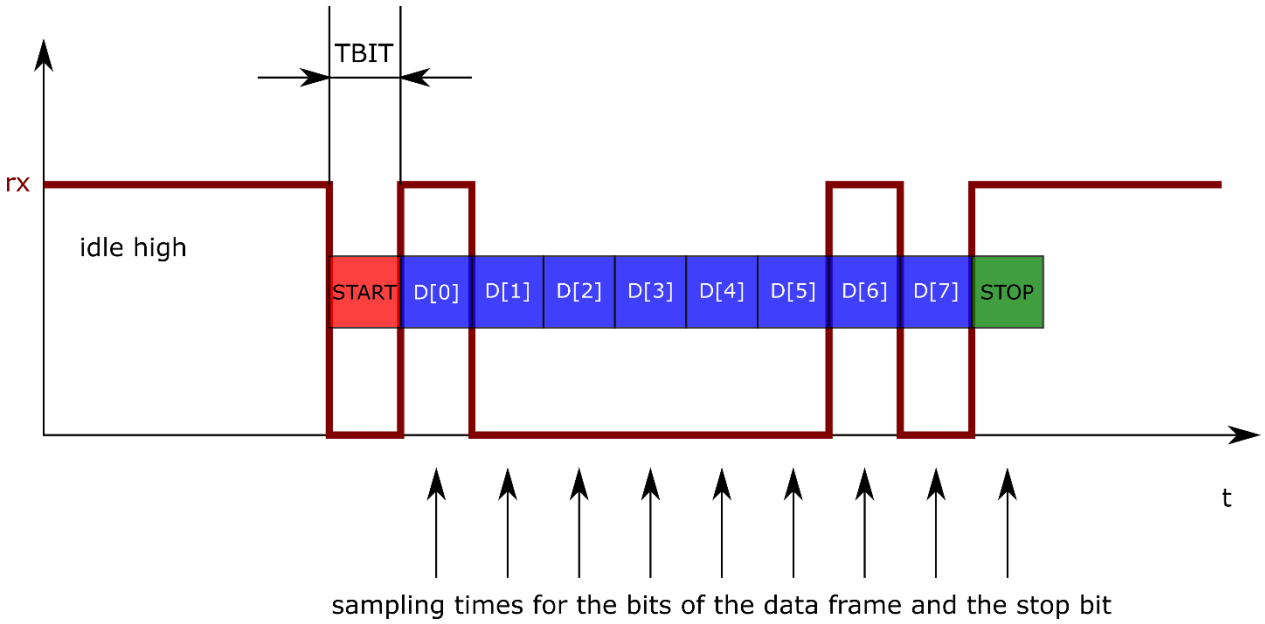

Figure 1: UART data frame.

Specification

Inputs and outputs

- rst_n: active low reset
- clk: clock → active edge = positive edge
- rx: uart rx input (One START bit, eight DATA bits (LSB first), one STOP bit). Polarity as indicated in Figure 1.
- rx_data[7:0]: the received byte
- rx_idle: shall be low during an active uart frame (i.e. during START, DATA and STOP bits)

- rx_ready: indicates that valid data is available at rx_data.
- rx_error: indicates a framing error (i.e. rx at least for one clock cycle low during STOP).

Figure 2: UART rx IOs.

Parameters

- fclk: indicates the clock frequency (default is 50 000 000).
- fbaud: indicates the selected uart baud rate (default is 115200).

Functional requirements:

- Active low reset (rst_n == 1'b0).
- After reset is released (rst_n == 1'b1) the rx unit shall be ready for an incoming data frame.
- The duration of a bit shall be derived from the parameters fclk and fbaud. A counter shall be used to generate the timing.
- Sampling of the data bits shall be done in the middle of a bit time. The sampling time can be derived from the parameters fclk and fbaud.
- rx_ready shall be set when the STOP bit is over. It is not important if there was a framing error or not. rx_ready shall be cleared as soon as a new frame is started.
- rx_error shall be set if the rx input is low at least for one clock cycle during the STOP bit. It shall be cleared when a new frame is started.
- rx_idle is low during a running frame (i.e. during START, DATA and STOP bits) otherwise it shall be high.

Tasks

- Create the folder structure
  - ./uart_rx/sim
  - ./uart_rx/src

Task 01 – Design [2cp]

- Design the module uart_rx in a file uart_rx.sv.
- Use a state machine to control the bit sequence.
- Use a counter to control the timing of a bit.
- Use a second counter to count the number of data bits.
- Use flip flops for rx_ready and rx_error and set and clear them using the state machine.

Task 02 – Verification [2cp]

- Create a testbench that verifies the correct behaviour of the design.

- Use the already existing uart_tx module to generate the bit stream for rx. Send valid data frames 0xa5, 0x5a, 0xff and 0x00 and check that the received data is ok.
- Use combinatorial logic to be able to override the rx file from the testbench such that a framing error can be provoked. That that the rx_error flag is set.

Task 03 – Documentation [1cp]

- Create a short summary report “doc_uart_rx.pdf” that shows the simulation result.
    - Show that your design fulfils the specification (verification).
    - Discuss why you chose your implementation method (advantages, disadvantages).