Exercise 09: UART

Universal Asynchronous Receiver/Transmitter

UART performs serial to parallel decoding of data (in the receiver) and parallel to serial encoding of data (in the transmitter). It is an industrial standard for serial communication and is used in a wide variety of systems.

The data between UART devices are transmitted over single data line one symbol at a time and the format is illustrated in the following diagram:

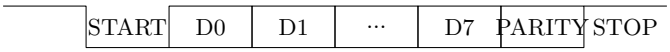

Figure 1: uart

At idle the data line is kept at high value (`'1'`). The transmission consists of a sequence of symbols that mark the start of transmission (START mark, '0' bit identified by a falling edge of data line), the preset number of data bits (typically from 5 to 8, with the value of the bit being the value of the data line) that are transmitted least significant bit first, optional parity bit, and stop bits (from 1 to 2 STOP marks with value of '1') that mark the end of transmission. The transmission rate (baud rate or symbols per second) is fixed with a typical value of 115200 often seen when communicating with serial terminals.

You are provided with an implementation of a transceiver `uart_tx` which can be configured to handle transmission with different number of data, stop and parity bits.

Implement uart receiver that handles serial stream configured with 1 stop bit and no parity bits.

Use the following interface:

```
entity uart_rx is
generic (
    g_DATA_BITS : positive := 8;
    -- clocks per symbol = positive(1000000.0*CLK_MHZ)/BAUD_RATE
    g_CpS : positive--;
);
port (
    -- serial data
    -- - idle is logic high
    -- - start bit is logic low
    -- - least significant bit first
    -- - stop bit is logic high
    i_sdata     : in    std_logic;

    -- parallel data interface
    o_rdata     : out   std_logic_vector(g_DATA_BITS-1 downto 0);
    i_rack      : in    std_logic;
    o_rempty    : out   std_logic;

    i_reset_n   : in    std_logic;
    i_clk       : in    std_logic--;
);
end entity;
```

Generic `g_DATA_BITS` specifies number of data bits in serial stream and `g_CpS` specifies number of clock cycles used to transmit one symbol (e.g. data bit).

Instructions:

- Use state machine to encode current serial data bit type (start, data, etc.)
- Use fifo to buffer decoded decoded data for subsequent readout by upstream (route read interface ports of fifo to parallel data interface ports)
- To detect start bit (and transition from idle state), monitor input serial stream for falling edge during idle state (going from idle `'1'` to start bit `'0'`)
- Assume that each bit is `g_CpS` clock cycles long and detect value of subsequent bits by sampling at the middle of each bit signal
- During data state, save bits from serial data stream into register and write to fifo when last bit is received

Test with the provided testbench and top file. Note that testbench uses `g_DATA_BITS := 4` and `g_CpS := 4`.

Questions:

- What is the maximum effective data rate for the above UART receiver (one start bit, 4 data bits and one stop bit)?
- What error conditions (in external input or internal state) are possible?

The labs take place in Staudingerweg 9 building: group 1 in room 4-516 on Thurday and group 2 in Newton room on Friday.

Send your solutions through MOODLE (https://moodle.uni-mainz.de).

The solution should include:

- Source code (only **your** `.vhd` files)
- Brief description if applicable, questions, etc.
- Screen shot of the simulation (use simulation range of 1 microsecond)
- Submit your files in one `.zip` archive
