RTL project description:

About Me:

My name is **Vignesh**, from India, and I am currently working at **CDAC** as an **RTL Design Engineer**. I have **3.8+ years of experience** in RTL design, specializing in microarchitecture, cache design, and verification.

Project: Advanced Supercomputing Research (ACR):

Project Duration: Ongoing
Project Title: Design of a 64-bit RISC-V Superscalar Out-of-Order CPU

Roles & Responsibilities:

Primary Tasks

Microarchitecture Definition:

- Designed **I-Cache** and **D-Cache** as configurable cache memories (dual-port RAM with basic cache structure including LRU).
- Developed **tag comparator**, **victim cache**, **branch predecode unit**, **MSHR**, **TLB**, **prefetcher**, and **way predictor**.

Block Specification & Component Design:

- Designed a pipelined **L1 I-Cache** and **L1 D-Cache** (cache size, ways, and cache line length are configurable) using **Verilog**.

- Implemented **MESI Protocol** supporting both L1 caches.
- Integrated cache architecture with **prefetch unit**, **way predictor**, **victim cache**, and **MSHR** to enhance cache performance.
- Developed an **AXI interface** for integrating the cache with main memory.

Verification:

- Verified each module using **Verilog testbenches** and performed coverage checks.
- Conducted **linting checks** for quality assurance before sign-off.

Synthesis:

- Performed synthesis using **Cadence Genus (28 nm, 2 GHz)** and analyzed synthesis results.
- Created detailed documentation for both **I-Cache** and **D-Cache**.

Automation:

- Developed automation scripts using **TCL** and **Shell** on Linux to streamline synthesis processes.

Tools Used:

- **Simulation:** Questa Sim-64 2019.2, Vivado 2019.1
- **Synthesis:** Cadence Genus
- **Linting:** Spyglass Lint

Work I have done:

- I cloned the repository https://github.com/alexforencich/verilog-uart and committed it **without modifications**, then pushed it into my repository on the main (master) branch.

- I created a branch named **"1-correct-error-in-uart-transmitter-uart_tx"**, linked to **Issue #1 – Correct error in UART Transmitter (uart_tx)**. In this branch, I specifically modified (added a bug) in the TX module. Along with the issue, I provided a specification for the TX module. Based on the specification and the buggy code, I expected the LLM to generate the corrected version of the code.

- I created another branch named **"3-correct-error-in-uart-receiver-uart_rx"**, linked to **Issue #2 – Correct error in UART Receiver (uart_rx)**. In this branch, I specifically modified (added a bug) in the RX module. Along with the issue, I provided a specification for the RX module. Based on the specification and the buggy code, I expected the LLM to generate the corrected version of the code.

- I then created a branch named **"5-complete-the-design-uart-top-module"**, linked to **Issue #5 –**

Complete the design of UART Top Module. In this branch, I specifically modified (added a bug) in the Top module. Along with the issue, I provided a specification for the Top module. Based on the specification and the buggy code, I expected the LLM to generate the corrected version of the code.

Description of the changes I made to break the UART tx, UART rx, uart modules.

In uart_tx - error i made:

```
    if (s_axis_tvalid) begin
      s_axis_tready_reg <= !s_axis_tready_reg;
      prescale_reg <= (prescale << 3)-1;
      bit_cnt <= DATA_WIDTH+1;
      data_reg <= {1'b1, s_axis_tdata};
      bit_cnt <= DATA_WIDTH;
      data_reg <= {s_axis_tdata,1'b0};
      txd_reg <= 0;
      busy_reg <= 1;
    end
  end else begin
    if (bit_cnt > 1) begin
      bit_cnt <= bit_cnt - 1;
      prescale_reg <= (prescale << 3)-1;
```

```
            {data_reg, txd_reg} <= {1'b0, data_reg};
        end else if (bit_cnt == 1) begin
            {txd_reg,data_reg} <= {1'b0, data_reg};
        end else if (bit_cnt == 0) begin
            bit_cnt <= bit_cnt - 1;
            prescale_reg <= (prescale << 3);
            prescale_reg <= (prescale << 3) -1;
            txd_reg <= 1;
        end
    end
```

I made critical errors in the following parts:

- **bit_cnt** – the overall bit count does not match the expected value.
- **data_reg** – the data is not stored in proper UART format.
- **Shift logic** – I introduced a bug in the left-shift logic that was supposed to ensure UART formatting. Because of this, the overall output becomes completely zero.
- **Prescale computation logic** – requires modification.
- **bit_cnt calculation logic** – requires correction as well.

Regarding the specification, I have provided a clear explanation in **Issue #1**:

https://github.com/vigneshs41/Vignesh_S_RTL_UART/issues/1

For a more detailed view of the code differences, please check this pull request:

"https://github.com/vigneshs41/Vignesh_S_RTL_UART/pull/2/files"

In uart_rx - error i made:

```
  if (prescale_reg > 0) begin
            prescale_reg <= prescale_reg - 1;
        end else if (bit_cnt > 0) begin
            if (bit_cnt > DATA_WIDTH+1) begin
                if (!rxd_reg) begin
                if (!rxd) begin
                    bit_cnt <= bit_cnt - 1;
                    prescale_reg <= (prescale << 3)-1;
                end else begin


          end else if (bit_cnt > 1) begin
                bit_cnt <= bit_cnt - 1;
                prescale_reg <= (prescale << 3)-1;
                data_reg <= {rxd_reg, data_reg[DATA_WIDTH-1:1]};
                data_reg <= {rxd, data_reg[DATA_WIDTH-1:0]};
            end else if (bit_cnt == 1) begin
                bit_cnt <= bit_cnt - 1;
                if (rxd_reg) begin
                if (rxd) begin
                    m_axis_tdata_reg <= data_reg;
                    m_axis_tvalid_reg <= 1;
                    overrun_error_reg <= m_axis_tvalid_reg;
                end else begin
                    overrun_error_reg <= ~m_axis_tvalid_reg;
                end
            end
            else begin
                    frame_error_reg <= 1;
                end
            end
        end else begin
            busy_reg <= 0;
            if (!rxd_reg) begin
                prescale_reg <= (prescale << 2)-2;
                bit_cnt <= DATA_WIDTH+2;
            if (!rxd) begin
                prescale_reg <= (prescale << 3)-1;
                bit_cnt <= DATA_WIDTH;
                data_reg <= 0;
                busy_reg <= 1;
            end
        end
```

I made the following errors:

- In **rxd**, instead of taking data from the register, I mistakenly took it directly from the input port, which causes a synchronous issue.
- In **data_reg** assignment, which results in improper data storage.
- In **bit_cnt** and **prescale_reg** computation, which leads to incorrect bit count calculation and baud rate computation.
- In **overrun_error_reg**, which results in incorrect error handling.

Regarding the specification, I have provided a clear explanation in the issue:

https://github.com/vigneshs41/Vignesh_S_RTL_UART/issues/3

For more detailed regrading code difference:
https://github.com/vigneshs41/Vignesh_S_RTL_UART/pull/4/files

---

In top module uart - error i made:

```
 .clk(clk),
  .rst(rst),
  // axi input
  .s_axis_tdata(s_axis_tdata),
  .s_axis_tvalid(s_axis_tvalid),
    .s_axis_tready(s_axis_tready),
    .s_axis_tdata(m_axis_tdata),
    .s_axis_tvalid(m_axis_tvalid),
    .s_axis_tready(m_axis_tready),
    // output
    .txd(txd),
    .txd(rxd),
    // status
    .busy(tx_busy),
    // configuration
    .prescale(prescale)
);

uart_rx #(
    .DATA_WIDTH(DATA_WIDTH)
    .DATA_WIDTH(DATA_WIDTH+1)
)
uart_rx_inst (
    .clk(clk),
    .rst(rst),
    .rst(~rst),
    // axi output
    .m_axis_tdata(m_axis_tdata),
    .m_axis_tvalid(m_axis_tvalid),
    .m_axis_tready(m_axis_tready),
    .m_axis_tdata(s_axis_tdata),
    .m_axis_tvalid(s_axis_tvalid),
    .m_axis_tready(s_axis_tready),
    // input
    .rxd(rxd),
    .rxd(txd),
```

From the top module, I made bugs in the **rst** signal, **DATA_WIDTH** assignment, as well as in the input and output port connections between the top module and TX/RX modules, which are completely misaligned.

Regarding the specification, I have provided a clear explanation in the issue:
https://github.com/vigneshs41/Vignesh_S_RTL_UART/issues/5
For more detailed regrading code difference:
https://github.com/vigneshs41/Vignesh_S_RTL_UART/pull/6/files

---