EE 705: VLSI DESIGN LAB 2025
PROJECT- UART-LITE MODULE

TAs: Shahrukh + Harshitha

Note:

1. Submission is **only** through Moodle in the form of a PDF file upload.
2. Keep the port names and module names as mentioned in the question (including the case—whatever is capital must be capital & whatever is small must be small).
3. Code must be in **Verilog**(case-sensitive)**.**

---

1) Design the uart module (module name: **uart_lite**) for a RISCV processor, that comprises of the following ports:
   a. Input Ports:
      i. clk_i : Clock signal for uart.
      ii. rst_i: Synchronous Active high Reset
      iii. cfg_awvalid_i: Valid Address signal requires for Handshaking to enable write
      iv. cfg_awaddr_i: Address of particular buffer to be written. (Some default value of address as flag to choose Tx-Mode, Rx-Mode, status and Control.)
      v. cfg_wvalid_i : Unused Signal, Keep it for Uniformity.
      vi. cfg_wdata_i : Configuration data and actual data to be sent by UART.
      vii. cfg_wstrb_i: Unused Signal, Keep it for Uniformity.
      viii. cfg_bready_i : For Hanshaking of response channel similar to AXI response channel.(you don't need to check error of response channel. Ie. cfg_bresp_o is always zero for simpilicity.)
      ix. cfg_arvalid_i : Handshaking signal to enable read and also affects cfg_awready_o.
      x. cfg_araddr_i : Address of particular buffer to be read.
      xi. cfg_rready_i: Handshaking signal required for read and also affects r_valid.
      xii. rx_i: UART serially received message input.
   b. Output Ports:
      i. cfg_awready_o: Handshaking signal required for write Enable and also affects cfg_wready_o.
      ii. cfg_wready_o : Handshaking signal required for write data.(core requires this.)
      iii. cfg_bvalid_o: Handshaking signal required for write response channel.
      iv. cfg_bresp_o: Response of write response channel.(keep it 2'b0)
      v. cfg_arready_o: Handshaking Signal for Read Address.(keep it low when cfg_rvalid_o is high)
      vi. cfg_rvalid_o: Handshaking signal for Read Data.
      vii. cfg_rdata_o: Data read from configuration buffers.
      viii. cfg_rresp_o: Signal to tell read response is OK. (keep it 2'b0)
      ix. tx_o: UART serially Transmitted message output.
      x. intr_o: Interrupt to generate after Successful Completion and if Receiver is ready to receive.

2) Functional Description of the UART-Lite module:
   a. Read about UART communication Protocol(see link). Ports description is Similar to AXI.(It will be better if You read AXI first then UART.
   b. IT has configuration Buffers {ULITE_RX, ULITE_TX, ULITE_STATUS, ULITE_CONTROL}. These buffer are handled by core and on the basis of these Values UART performs its Operation.
   c. First of Handle read_enable and write_enable signals.
   d. When cfg_araddr_i is `ULITE_RX and read_en_w is high It is in RX-mode and gets reset.(reset data state).
   e. When cfg_awaddr_i is `ULITE_TX and write_en_w is high, it will Buffer the data(msg) to transmit.
   f. When cfg_awaddr_i is `ULITE_STATUS it is only important to read the current status of required flag or control signals.
   g. When cfg_awaddr_i is `ULITE_CONTROL it writes the cfg_wdata_i on ulite_control_rst_tx and ulite_control_rst_rx.
3) Make a test bench to test the complete module, showing all the input and output signals properly
4) The Hardware needs to be tested on FPGA using VIO and ILA.
5) Ensure that your complete module is compliant with RISCV ISA (RV32I)
6) Draw a block diagram. It should clearly represent how the integration of the UART-Lite module will happen with the other modules.

Understanding the UART Lite Register Definitions
Here's a breakdown of the **UART Lite** registers and their functions:

<u>1. ULITE_RX (Receive Register)</u>

- Address: 0x00
- **Bits [7:0] (ULITE_RX_DATA)** → Holds received data (8-bit).
- Read this register to get received data from the RX FIFO.

<u>2. ULITE_TX (Transmit Register)</u>

- Address: 0x04
- **Bits [7:0] (ULITE_TX_DATA)** → Holds data to be transmitted (8-bit).
- Write data to this register to send it via UART.

<u>3. ULITE_STATUS (Status Register)</u>

- Address: 0x08
- **Bit 4 (ULITE_STATUS_IE)** → Interrupt Enable Status.
- **Bit 3 (ULITE_STATUS_TXFULL)** → TX FIFO full.
- **Bit 2 (ULITE_STATUS_TXEMPTY)** → TX FIFO empty.
- **Bit 1 (ULITE_STATUS_RXFULL)** → RX FIFO full.
- **Bit 0 (ULITE_STATUS_RXVALID)** → RX FIFO has valid data.

<u>4. ULITE_CONTROL (Control Register)</u>

- Address: 0x0C
- **Bit 4 (ULITE_CONTROL_IE)** → Enable UART interrupts.
- **Bit 1 (ULITE_CONTROL_RST_RX)** → Reset RX FIFO.

- **Bit 0 (ULITE_CONTROL_RST_TX)** → Reset TX FIFO.

Here is an Example to used definition:

```
 `define ULITE_CONTROL    8'hc

`define ULITE_CONTROL_IE      4
`define ULITE_CONTROL_IE_DEFAULT    0
`define ULITE_CONTROL_IE_B         4
`define ULITE_CONTROL_IE_T         4
`define ULITE_CONTROL_IE_W          1
`define ULITE_CONTROL_IE_R         4:4

`define ULITE_CONTROL_RST_RX      1
`define ULITE_CONTROL_RST_RX_DEFAULT    0
`define ULITE_CONTROL_RST_RX_B         1
`define ULITE_CONTROL_RST_RX_T         1
`define ULITE_CONTROL_RST_RX_W          1
`define ULITE_CONTROL_RST_RX_R         1:1

`define ULITE_CONTROL_RST_TX      0
`define ULITE_CONTROL_RST_TX_DEFAULT    0
`define ULITE_CONTROL_RST_TX_B         0
`define ULITE_CONTROL_RST_TX_T         0
`define ULITE_CONTROL_RST_TX_W          1
`define ULITE_CONTROL_RST_TX_R         0:0
```

ADDITIONAL Materials and Hints:

a. https://www.youtube.com/watch?v=sTHckUyxwp8
b. cs.sfu.ca/~ashriram/Courses/CS295/assets/notebooks/RISCV/RISCV_CARD.pdf
c. Include this in your code:

```
module uart_lite
(
   // Inputs
    input         clk_i
   ,input         rst_i
   ,input         cfg_awvalid_i
   ,input  [31:0]  cfg_awaddr_i
   ,input         cfg_wvalid_i
   ,input  [31:0]  cfg_wdata_i
   ,input  [3:0]   cfg_wstrb_i
   ,input         cfg_bready_i
   ,input         cfg_arvalid_i
   ,input  [31:0]  cfg_araddr_i
   ,input         cfg_rready_i
   ,input         rx_i
```

```
        // Outputs
        ,output        cfg_awready_o
        ,output        cfg_wready_o
        ,output        cfg_bvalid_o
        ,output [1:0]   cfg_bresp_o
        ,output        cfg_arready_o
        ,output        cfg_rvalid_o
        ,output [31:0]  cfg_rdata_o
        ,output [1:0]   cfg_rresp_o
        ,output        tx_o
        ,output        intr_o
      );

      //This always block can be part of UART-Lite , It is given for your understanding.

  always @( *)
  begin
     data_r = 32'b0;

     case (cfg_araddr_i[7:0])
     `ULITE_RX:
     begin
        data_r[`ULITE_RX_DATA_R] = ulite_rx_data_in_w; //msg data coming serially
     end
     `ULITE_STATUS:
     begin
        data_r[`ULITE_STATUS_IE_R] = ulite_status_ie_in_w;
        data_r[`ULITE_STATUS_TXFULL_R] = ulite_status_txfull_in_w;
        data_r[`ULITE_STATUS_TXEMPTY_R] = ulite_status_txempty_in_w;
        data_r[`ULITE_STATUS_RXFULL_R] = ulite_status_rxfull_in_w;
        data_r[`ULITE_STATUS_RXVALID_R] = ulite_status_rxvalid_in_w;
     end
     `ULITE_CONTROL:
     begin
        data_r[`ULITE_CONTROL_IE_R] = ulite_control_ie_q;
     end
     default : data_r = 32'b0;
     endcase
      end
```