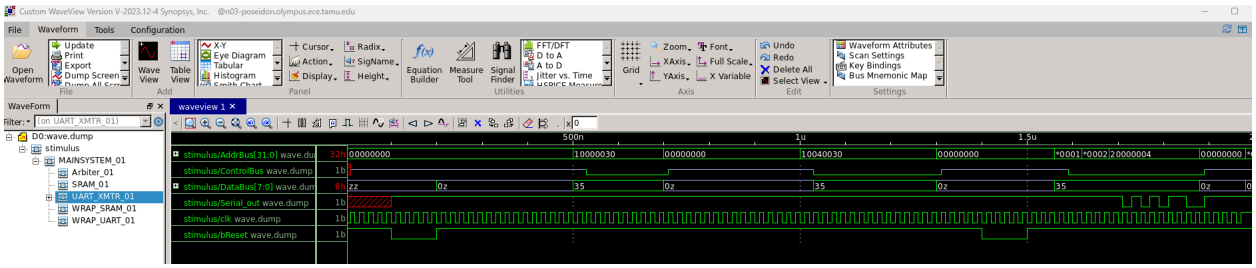
1. Screenshots of the waveform screenshot after functional simulation.



2. Justification of the correctness of the results.

The testbench first writes the value 35 to SRAM address 18'h30. Later, it reads from the same address, and it can be observed that the data value read out is 35. After the SRAM is read, the data value 35 is loaded onto the data bus and subsequently to the UART transmitter. The waveform shows that the Serial_out signal's output is 0101011001, which, when reversed, is {1, 8'h35, 0}, where 0 and 1 correspond to the start and stop bits, respectively. This confirms that our system is functionally correct for SRAM read, write, and UART read and write.

3. Screenshots of the code you implemented.

WRAP_SRAM.v

```
assign ControlBus = IntEnable? 1'b0 : 1'bz;
// You need to assign DataBus here using continuous assign statement.
//..
assign DataBus = (IntEnable && (AddressBus[31:28] == 4'b0001) && AddressBus[18]) ? OutData : 1'bz;

always @(posedge clk)
begin
      if(IntEnable==1'b1)              // Bus free
      begin
          if(AddressBus[31:28] == 4'b0001) // Address Decoding Matching
          begin
           // Insert your code here
           // ..
           // ..
           Address <= AddressBus[17:0];
           InData  <= DataBus;
           bCE     <= AddressBus[19];
           bWE     <= AddressBus[18];
          end
      end
end
```

WRAP_UART.v

```
assign ControlBus = IntEnable ? 1'b0 : 1'bz;

always @(posedge clk)
begin
        if(IntEnable)                   // Bus Free
        begin
            if(AddressBus[31:28] == 4'b0010) // Address Decoding Matching
            begin
             // Insert your code here
             // ...
             // ...
             InData           <= DataBus;
             Load_XMT_datareg <= AddressBus[0];
             Byte_ready       <= AddressBus[1];
             T_byte           <= AddressBus[2];
            end
        end
    else
    begin
                T_byte <= 1'b0;
            Byte_ready <= 1'b0;
            Load_XMT_datareg <= 1'b0;
            InData <= 8'h00;
    end
end
```

I also modified tb.v as I was facing issues while generating the wave.dump file.

Tb.v

```
    initial
    begin
        $dumpfile ("wave.dump");
        $dumpvars (0, stimulus);

        $dumpvars (2, Serial_out);
        $dumpvars (2, clk);
        $dumpvars (2, Breq);
        $dumpvars (2, Bgnt);
        $dumpvars (2, DataBus);
        $dumpvars (2, AddrBus);
        $dumpvars (2, ControlBus);
        $dumpvars (2, bReset);
    end
```