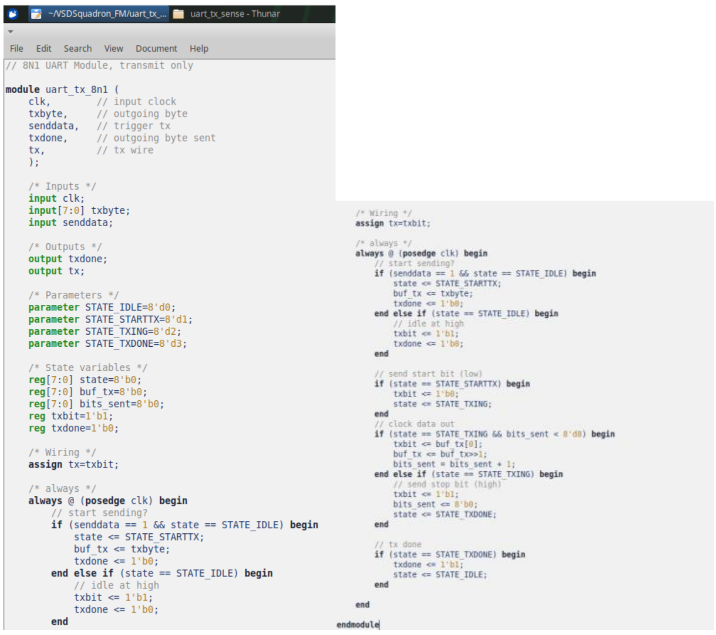
Task 4

The analysis,


~/VSDSquadron_FM/uart_tx_...
uart_tx_sense - Thunar
File Edit Search View Document Help

```verilog
// 8N1 UART Module, transmit only

module uart_tx_8n1 (
    clk,        // input clock
    txbyte,     // outgoing byte
    senddata,   // trigger tx
    txdone,     // outgoing byte sent
    tx,         // tx wire
    );

    /* Inputs */
    input clk;
    input[7:0] txbyte;
    input senddata;

    /* Outputs */
    output txdone;
    output tx;

    /* Parameters */
    parameter STATE_IDLE=8'd0;
    parameter STATE_STARTTX=8'd1;
    parameter STATE_TXING=8'd2;
    parameter STATE_TXDONE=8'd3;

    /* State variables */
    reg[7:0] state=8'b0;
    reg[7:0] buf_tx=8'b0;
    reg[7:0] bits_sent=8'b0;
    reg txbit=1'b1;
    reg txdone=1'b0;

    /* Wiring */
    assign tx=txbit;

    /* always */
    always @ (posedge clk) begin
        // start sending?
        if (senddata == 1 && state == STATE_IDLE) begin
            state <= STATE_STARTTX;
            buf_tx <= txbyte;
            txdone <= 1'b0;
        end else if (state == STATE_IDLE) begin
            // idle at high
            txbit <= 1'b1;
            txdone <= 1'b0;
        end
```

```verilog
    /* Wiring */
    assign tx=txbit;

    /* always */
    always @ (posedge clk) begin
        // start sending?
        if (senddata == 1 && state == STATE_IDLE) begin
            state <= STATE_STARTTX;
            buf_tx <= txbyte;
            txdone <= 1'b0;
        end else if (state == STATE_IDLE) begin
            // idle at high
            txbit <= 1'b1;
            txdone <= 1'b0;
        end
```

```verilog
        // send start bit (low)
        if (state == STATE_STARTTX) begin
            txbit <= 1'b0;
            state <= STATE_TXING;
        end
        // clock data out
        if (state == STATE_TXING && bits_sent < 8'd8) begin
            txbit <= buf_tx[0];
            buf_tx <= buf_tx>>1;
            bits_sent = bits_sent + 1;
        end else if (state == STATE_TXING) begin
            // send stop bit (high)
            txbit <= 1'b1;
            bits_sent <= 8'b0;
            state <= STATE_TXDONE;
        end

        // tx done
        if (state == STATE_TXDONE) begin
            txdone <= 1'b1;
            state <= STATE_IDLE;
        end

    end

endmodule
```

This system is made up of 3 major components,

The data buffer management

UART protocol control

Transmission control protocol

Here,

The data buffer stores incoming data, provides stability during transmission and handles synchronization

The transmission controller manages bit by bit transmission, UART protocol timing and handles the start/stop bit generation.

The sensor data arrives with valid signal assertion which is captured at the IDLE state

START: generates the UART start bit

DATA: transmits 8 bits sequentially

STOP: ensure proper termination with the high bit

Ready accepts new data

Tx_out gives continuous UART stream

The state transitions ensure reliable data transfer

The basic workflow is,

When senddata = 1, transmission starts

The start bit is sent to signal the beginning

Then the transmission of 8 bits occurs

The stop bit is sent at last

The tx done = 1 indicates completion

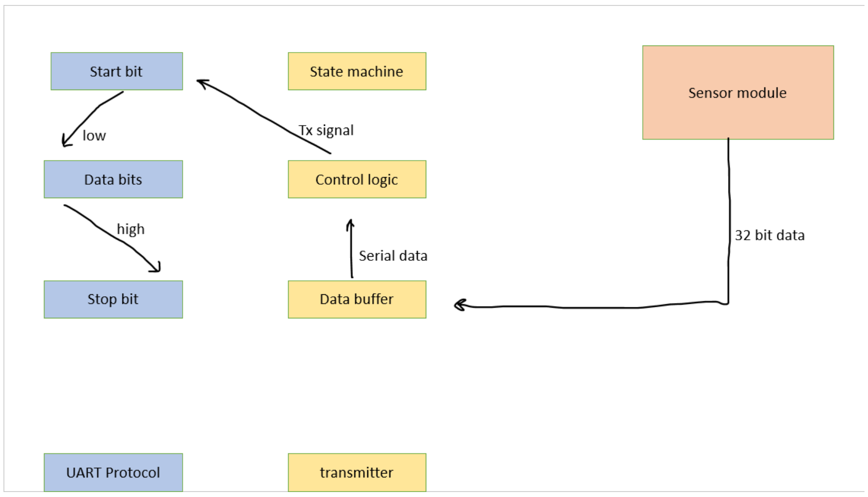
The IDLE mode is set as the machine waits for new data

The state machine signals,

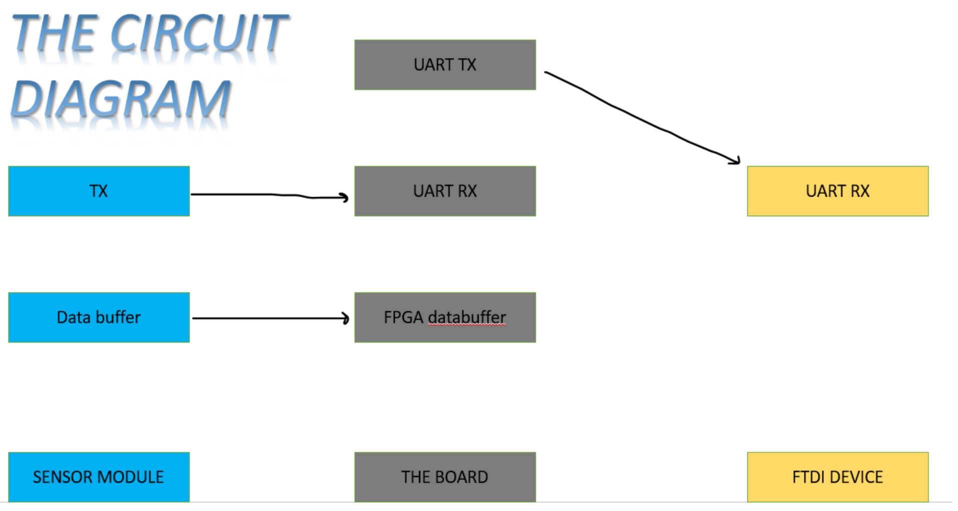
- STATE_IDLE: Doing nothing (waiting for data).
- STATE_STARTTX: Sends start bit (0) and prepares to send data.
- STATE_TXING: Sends 8 bits one by one (LSB first).
- STATE_TXDONE: Sends stop bit (1), then signals txdone = 1.

If senddata is 1 it stores byte in buf_tx and moves to the STATE_STARTTX else it stays IDLE.

The block diagram



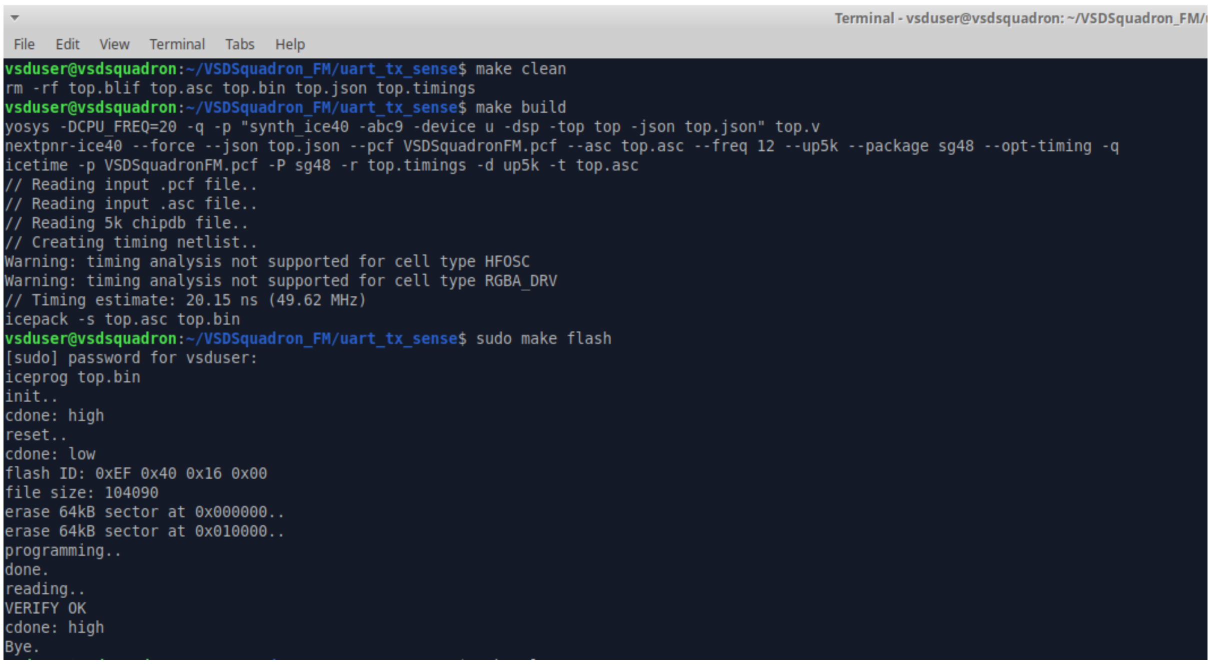
The circuit diagram,



FOR STEP 3, we must create the files,

- Makefile
- Top.v
- Uart_trx.v
- VSDSquadron.pcf

Then we may flash the code to the board as shown in the image.


Terminal - vsduser@vsdsquadron: ~/VSDSquadron_FM/
File Edit View Terminal Tabs Help

```
vsduser@vsdsquadron:~/VSDSquadron_FM/uart_tx_sense$ make clean
rm -rf top.blif top.asc top.bin top.json top.timings
vsduser@vsdsquadron:~/VSDSquadron_FM/uart_tx_sense$ make build
yosys -DCPU_FREQ=20 -q -p "synth_ice40 -abc9 -device u -dsp -top top -json top.json" top.v
nextpnr-ice40 --force --json top.json --pcf VSDSquadronFM.pcf --asc top.asc --freq 12 --up5k --package sg48 --opt-timing -q
icetime -p VSDSquadronFM.pcf -P sg48 -r top.timings -d up5k -t top.asc
// Reading input .pcf file..
// Reading input .asc file..
// Reading 5k chipdb file..
// Creating timing netlist..
Warning: timing analysis not supported for cell type HFOSC
Warning: timing analysis not supported for cell type RGBA_DRV
// Timing estimate: 20.15 ns (49.62 MHz)
icepack -s top.asc top.bin
vsduser@vsdsquadron:~/VSDSquadron_FM/uart_tx_sense$ sudo make flash
[sudo] password for vsduser:
iceprog top.bin
init..
cdone: high
reset..
cdone: low
flash ID: 0xEF 0x40 0x16 0x00
file size: 104090
erase 64kB sector at 0x000000..
erase 64kB sector at 0x010000..
programming..
done.
reading..
VERIFY OK
cdone: high
Bye.
```

Then we install picocom using

- sudo apt install picocom

then we connect the board and enter the command

- make terminal

to verify the outcome (continuous Ds).

This is shown in the following video

20250402-0659-39.6938647.mp4

The LED glows red in the FPGA board

Therefore task 4 is successfully completed.

---------presented by Archana Bhat-----