Name: Rahul J. Teradal

UART Project Implementation

1) UART Transmitter (TX) Code — Verilog.

```
module uart_tx (
  input        clk,
  input        rst,
  input        tx_start,
  input  [7:0] tx_data,
  output reg   tx,
  output reg   tx_busy
);
parameter CLKS_PER_BIT = 1667;  // Adjust according to your baud rate and clk
freq
reg [7:0] data_reg;
reg [3:0] bit_index;
reg [15:0] clk_count;
reg       sending;
reg       start_latched;

always @(posedge clk or posedge rst) begin
  if (rst) begin
    tx <= 1'b1;
    tx_busy <= 0;
    clk_count <= 0;
    bit_index <= 0;
    sending <= 0;
    data_reg <= 0;
  end else begin
    if (tx_start && !tx_busy && !sending) begin
      data_reg <= tx_data;
      sending <= 1;
      tx_busy <= 1;
      clk_count <= 0;
      bit_index <= 0;
    end else if (sending) begin
      if (clk_count == CLKS_PER_BIT - 1) begin
        clk_count <= 0;
    bit_index <= bit_index + 1;
   end else begin
    clk_count <= clk_count + 1;
   end

   case (bit_index)
    0:  tx <= 0;            // Start bit
    1:  tx <= data_reg[0];
    2:  tx <= data_reg[1];
    3:  tx <= data_reg[2];
    4:  tx <= data_reg[3];
    5:  tx <= data_reg[4];
    6:  tx <= data_reg[5];
    7:  tx <= data_reg[6];
    8:  tx <= data_reg[7];
    9:  tx <= 1;            // Stop bit
    10: begin
     sending <= 0;
     tx_busy <= 0;
    end
   endcase
  end
 end
end
endmodule
```

2) UART Transmitter Testbench

```
`timescale 1ns/1ps
module uart_tx_tb;
reg clk = 0;
reg rst = 1;
reg tx_start = 0;
reg [7:0] tx_data = 8'b0;
wire tx;
wire tx_busy;
parameter CLK_PERIOD = 62.5;  // 100 MHz clock
uart_tx #(.CLKS_PER_BIT(1667□)) DUT (
 .clk(clk),
 .rst(rst),
 .tx_start(tx_start),
 .tx_data(tx_data),
 .tx(tx),
 .tx_busy(tx_busy)
);
```

```
// Clock generation
always #(CLK_PERIOD/2) clk = ~clk;

initial begin
  $dumpfile("uart_tx_tb.vcd");
  $dumpvars(0, uart_tx_tb);

  // Initial reset
  #50;
  rst = 0;

  // Send 0xA5
  wait(!tx_busy);
  tx_data = 8'hA5;
  tx_start = 1;
  #CLK_PERIOD;
  tx_start = 0;

  // Wait for transmission to finish
  wait(!tx_busy);

  // Send 0x3C
  #100; // wait some idle time
  tx_data = 8'h3C;
  tx_start = 1;
  #CLK_PERIOD;
  tx_start = 0;

  wait(!tx_busy);

  // Finish
  #200;
  $finish;
end
endmodule
```

3) UART Receiver (RX) Code — Verilog

```
module uart_rx (
  input        clk,
  input        rst,
  input        rx,         // Serial input
  output reg   rx_done,
  output reg [7:0] rx_data,
  output reg   rx_busy
);
```

```
parameter CLKS_PER_BIT = 1667;
reg [15:0] clk_count = 0;
reg [3:0] bit_index = 0;
reg [7:0] rx_shift = 0;
reg       receiving = 0;
reg       rx_d1 = 1, rx_d2 = 1;

// Synchronize RX input
always @(posedge clk) begin
  rx_d1 <= rx;
  rx_d2 <= rx_d1;
end

always @(posedge clk or posedge rst) begin
  if (rst) begin
    rx_done    <= 0;
    rx_data    <= 0;
    rx_busy    <= 0;
    clk_count  <= 0;
    bit_index  <= 0;
    receiving  <= 0;
  end else begin
    rx_done <= 0;

    if (!receiving && rx_d2 == 0) begin
      // Detected start bit
      receiving <= 1;
      rx_busy   <= 1;
      clk_count <= CLKS_PER_BIT / 2;  // Align to mid-bit
      bit_index <= 0;
    end else if (receiving) begin
      if (clk_count == CLKS_PER_BIT - 1) begin
        clk_count <= 0;
        bit_index <= bit_index + 1;

        case (bit_index)
          0: ; // Start bit - ignore
          1: rx_shift[0] <= rx_d2;
          2: rx_shift[1] <= rx_d2;
          3: rx_shift[2] <= rx_d2;
          4: rx_shift[3] <= rx_d2;
          5: rx_shift[4] <= rx_d2;
          6: rx_shift[5] <= rx_d2;
          7: rx_shift[6] <= rx_d2;
          8: rx_shift[7] <= rx_d2;
          9: begin
      rx_data   <= rx_shift;
      rx_done   <= 1;
      rx_busy   <= 0;
      receiving <= 0;
     end
    endcase
   end else begin
    clk_count <= clk_count + 1;
   end
  end
 end
end
endmodule
```

4) UART Receiver Testbench

```
`timescale 1ns / 1ps
module uart_rx_tb;
 reg clk = 0;
 reg rst = 0;
 reg rx = 1;              // Idle is high
 wire rx_done;
 wire [7:0] rx_data;
 wire rx_busy;

 // Clock generation (60 MHz -> 16.67 ns period)
 always #8.33 clk = ~clk;

 parameter CLKS_PER_BIT = 1667;
 uart_rx #(.CLKS_PER_BIT(CLKS_PER_BIT)) uut (
  .clk(clk),
  .rst(rst),
  .rx(rx),
  .rx_done(rx_done),
  .rx_data(rx_data),
  .rx_busy(rx_busy)
 );

 // Task to send a UART frame: 1 start bit, 8 data bits, 1 stop bit
 task uart_send_byte;
  input [7:0] data;
  integer i;
  begin
   // Start bit
   rx = 0;
   #(CLKS_PER_BIT * 20);
```

```
    // Data bits (LSB first)
    for (i = 0; i < 8; i = i + 1) begin
      rx = data[i];
      #(CLKS_PER_BIT * 20);
    end

    // Stop bit
    rx = 1;
    #(CLKS_PER_BIT * 20);
  end
endtask

initial begin
  // Initial conditions
  rx = 1;
  rst = 1;
  #100;
  rst = 0;

  // Send a byte
  #10000;
  uart_send_byte(8'hA5);  // Binary: 10100101

  // Wait and observe
  #100000;

  if (rx_done) begin
    $display("Received byte: %h", rx_data);
  end else begin
    $display("Reception failed or not completed.");
  end

  $stop;
end
endmodule
```

5) UART Top Module (UART Loopback)

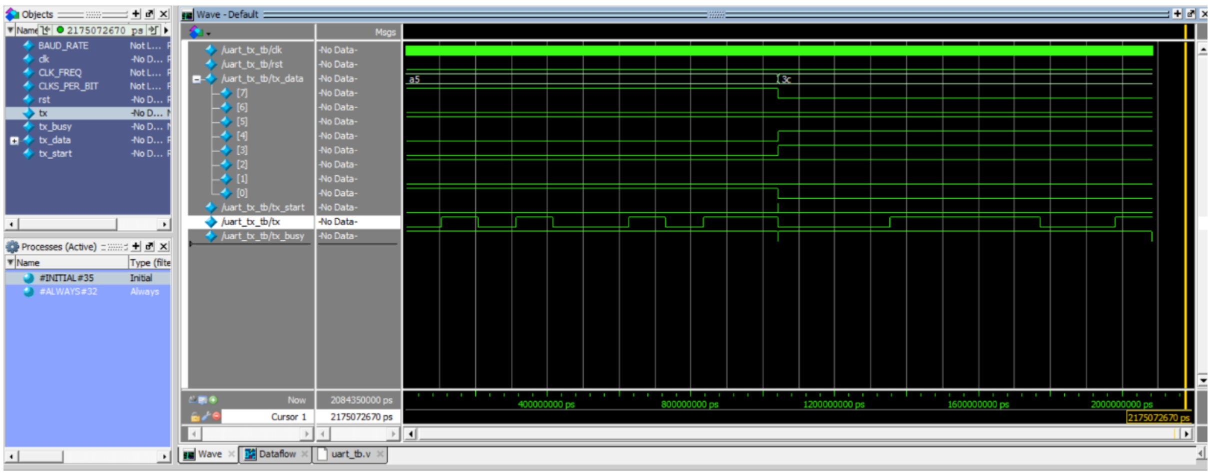
```
module uart_top (
  input        clk,
  input        rst,
  input        tx_start,
  input  [7:0] tx_data,
  output [7:0] rx_data,
  output       rx_done,
  output       tx_busy,
  output       rx_busy
);

wire tx_line;
uart_tx tx_inst (
  .clk(clk),
  .rst(rst),
  .tx_start(tx_start),
  .tx_data(tx_data),
  .tx(tx_line),
  .tx_busy(tx_busy)
);
uart_rx rx_inst (
  .clk(clk),
  .rst(rst),
  .rx(tx_line),       // Loopback connection
  .rx_done(rx_done),
  .rx_data(rx_data),
  .rx_busy(rx_busy)
);
endmodule
```

6) UART Testbench

```
`timescale 1ns/1ps
module uart_tb;

  // Clock and Reset
  reg clk = 0;
  reg rst = 1;

  // TX Interface
  reg        tx_start = 0;
  reg [7:0]  tx_data = 8'h00;
  wire       tx;
  wire       tx_busy;

  // RX Interface
  wire       rx_done;
  wire [7:0] rx_data;
  wire       rx_busy;

  // Generate 50MHz Clock (20ns period)
  always #10 clk = ~clk;

  // Instantiate TX module
  uart_tx #(.CLKS_PER_BIT(32'd16)) tx_inst (
    .clk(clk),
    .rst(rst),
    .tx_start(tx_start),
    .tx_data(tx_data),
    .tx(tx),
    .tx_busy(tx_busy)
  );

  // Instantiate RX module (connect RX to TX line)
  uart_rx #(.CLKS_PER_BIT(32'd16)) rx_inst (
    .clk(clk),
    .rst(rst),
    .rx(tx),
    .rx_done(rx_done),
    .rx_data(rx_data),
    .rx_busy(rx_busy)
  );

  // Stimulus
  initial begin
    // Initial reset
    #50;
    rst = 0;

    // Wait a bit before sending
    #100;

    // Send byte A5
    tx_data = 8'hA5;
    tx_start = 1;
    #20;
    tx_start = 0;

    // Wait for RX to finish
    wait (rx_done);
    $display("Received Data: %h", rx_data);

    // Wait and finish
    #200;
    $stop;
  end
endmodule
```

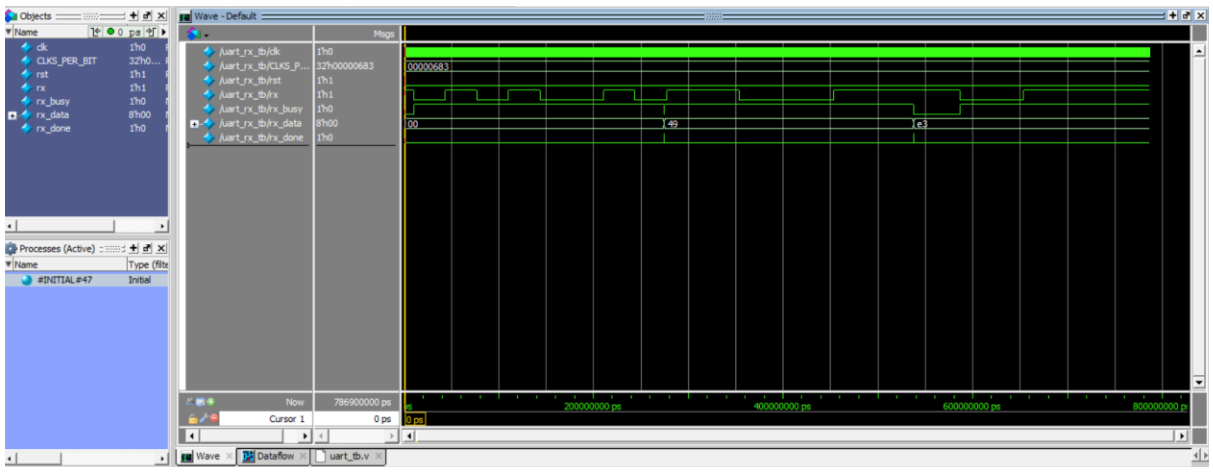
WAVE FORMS

1) UART Transmitter



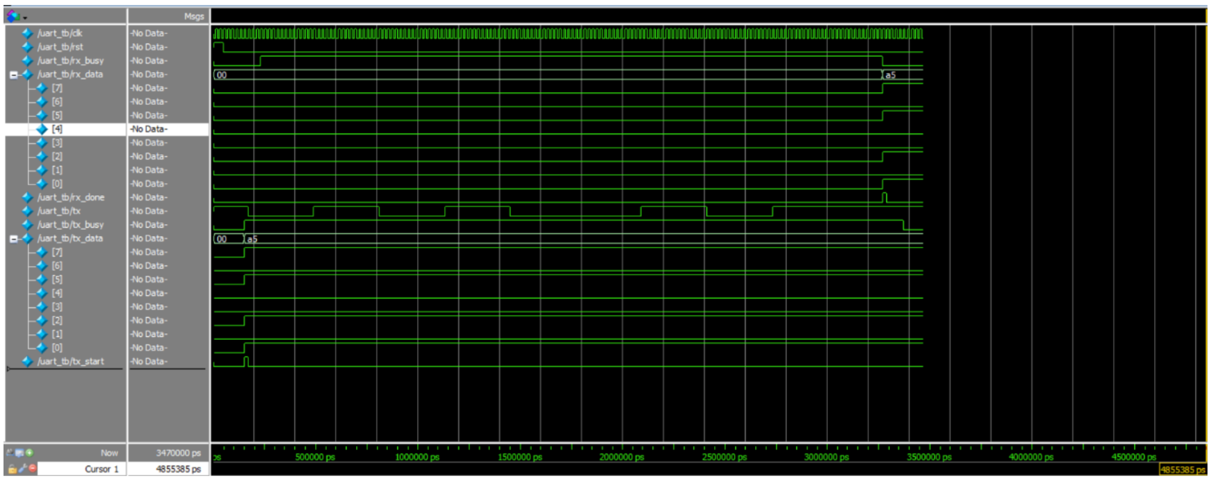
```
Name    2175072670 ps
BAUD_RATE     Not L...
clk           -No D...
CLK_FREQ      Not L...
CLKS_PER_BIT  Not L...
rst           -No D...
tx            -No D...
tx_busy       -No D...
tx_data       -No D...
tx_start      -No D...

Name          Type (filte
#INITIAL#35   Initial
#ALWAYS#32    Always

/uart_tx_tb/clk        -No Data-
/uart_tx_tb/rst        -No Data-
/uart_tx_tb/tx_data    -No Data-
  [7]                  -No Data-
  [6]                  -No Data-
  [5]                  -No Data-
  [4]                  -No Data-
  [3]                  -No Data-
  [2]                  -No Data-
  [1]                  -No Data-
  [0]                  -No Data-
/uart_tx_tb/tx_start   -No Data-
/uart_tx_tb/tx         -No Data-
/uart_tx_tb/tx_busy    -No Data-

Now        2084350000 ps
Cursor 1   2175072670 ps
```

2) UART Receiver

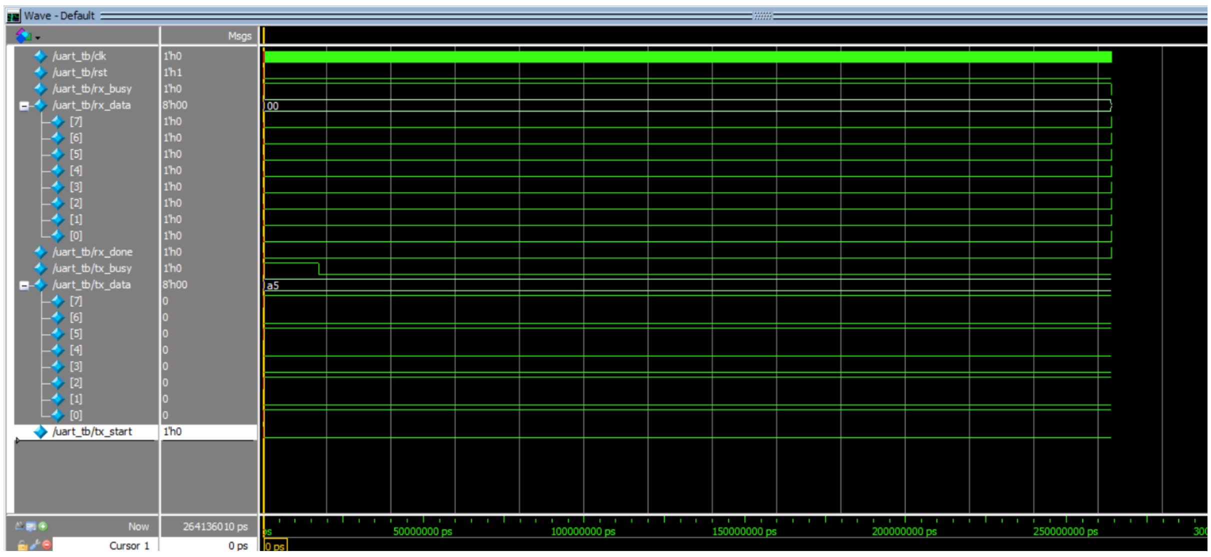


```
Name    0 ps
clk           1'h0
CLKS_PER_BIT  32'h0...
rst           1'h1
rx            1'h1
rx_busy       1'h0
rx_data       8'h00
rx_done       1'h0

Name          Type (filte
#INITIAL#47   Initial

/uart_rx_tb/clk         1'h0
/uart_rx_tb/CLKS_P...   32'h00000683
/uart_rx_tb/rst         1'h1
/uart_rx_tb/rx          1'h1
/uart_rx_tb/rx_busy     1'h0
/uart_rx_tb/rx_data     8'h00
/uart_rx_tb/rx_done     1'h0

Now        786900000 ps
Cursor 1   0 ps
```

3) UART Top



	Msgs
/uart_tb/clk	-No Data-
/uart_tb/rst	-No Data-
/uart_tb/rx_busy	-No Data-
/uart_tb/rx_data	-No Data-
[7]	-No Data-
[6]	-No Data-
[5]	-No Data-
[4]	-No Data-
[3]	-No Data-
[2]	-No Data-
[1]	-No Data-
[0]	-No Data-
/uart_tb/rx_done	-No Data-
/uart_tb/tx	-No Data-
/uart_tb/tx_busy	-No Data-
/uart_tb/tx_data	-No Data-
[7]	-No Data-
[6]	-No Data-
[5]	-No Data-
[4]	-No Data-
[3]	-No Data-
[2]	-No Data-
[1]	-No Data-
[0]	-No Data-
/uart_tb/tx_start	-No Data-
Now	3470000 ps
Cursor 1	4855385 ps



Wave - Default

	Msgs
/uart_tb/clk	1'h0
/uart_tb/rst	1'h1
/uart_tb/rx_busy	1'h0
/uart_tb/rx_data	8'h00
[7]	1'h0
[6]	1'h0
[5]	1'h0
[4]	1'h0
[3]	1'h0
[2]	1'h0
[1]	1'h0
[0]	1'h0
/uart_tb/rx_done	1'h0
/uart_tb/tx_busy	1'h0
/uart_tb/tx_data	8'h00
[7]	0
[6]	0
[5]	0
[4]	0
[3]	0
[2]	0
[1]	0
[0]	0
/uart_tb/tx_start	1'h0
Now	264136010 ps
Cursor 1	0 ps