**Course Code: EE461 Lab**

**Lab -08**

**PREPARED BY**

**Khandoker Samiul Hoque**

**Student Id: 19837**

**Github:**

**Exercise:**

**Transmitter Verilog Code:**

**Module:**

module UARTTx(clk,rst\_i,byteReady\_i,load\_i,TxByte\_i,busData\_i,serialOut\_o);

input clk;

input rst\_i;

input byteReady\_i;

input load\_i;

input TxByte\_i;

input[7:0] busData\_i;

output serialOut\_o;

parameter pIdle=2'b00;

parameter pWait=2'b01;

parameter pSend=2'b10;

reg[1:0] curSt\_r;

reg[1:0] nxtSt\_r;

reg loadData\_r;

reg start\_r;

reg clr\_r;

reg shift\_r;

reg[3:0] cnt\_r;

reg[7:0] busReg\_r;

reg[8:0] dataReg\_r;

//Sequential Logic

always@(posedge clk)begin

if(rst\_i)begin

curSt\_r <= pIdle;

end

else begin

curSt\_r <= nxtSt\_r ;

end

end

// Combo Logic

always@(\*)begin

loadData\_r=1'b0;

start\_r=1'b0;

clr\_r=1'b0;

shift\_r=1'b0;

nxtSt\_r= curSt\_r;

case(curSt\_r)

pIdle: begin

if(byteReady\_i)begin

loadData\_r=1'b1;

nxtSt\_r=pWait;

end

end

pWait: begin

if(TxByte\_i)begin

start\_r = 1'b1;

nxtSt\_r= pSend;

end

end

pSend: begin

if(cnt\_r ==9)begin

clr\_r = 1'b1;

nxtSt\_r= pIdle;

end

else begin

shift\_r = 1'b1;

end

end

default: begin

nxtSt\_r= pIdle;

end

endcase

end

//Implementation block- bit counter

always@(posedge clk)begin

if(rst\_i)begin

cnt\_r<=4'b0000;

end

else if(clr\_r)begin

cnt\_r<=4'b0000;

end

else if(shift\_r)begin

cnt\_r<= cnt\_r + 1'b1;

end

end

always@(posedge clk)begin

if(rst\_i)begin

busReg\_r <=8'b0000\_0000;

dataReg\_r<=9'b1\_1111\_1111;

end

else if(load\_i)begin

busReg\_r <= busData\_i;

end

else if(loadData\_r)begin

dataReg\_r <= {busReg\_r, 1'b1};

end

else if(start\_r)begin

dataReg\_r[0]<=1'b0; //output startbit ==0;

end

else if(shift\_r)begin

dataReg\_r<={1'b1, dataReg\_r[8:1] };

end

end

assign serialOut\_o = dataReg\_r[0];

endmodule

**Testbench:**

module tb();

reg clk;

reg rst\_i;

reg byteReady\_i;

reg load\_i;

reg TxByte\_i;

reg [7:0] busData\_i;

wire serialOut\_o;

UARTTx lab08 (.clk(clk), .rst\_i(rst\_i), .byteReady\_i(byteReady\_i),.load\_i(load\_i),.TxByte\_i(TxByte\_i),.busData\_i(busData\_i),.serialOut\_o(serialOut\_o));

initial begin

$dumpfile("dump.vcd");

$dumpvars;

clk = 0;

rst\_i = 1'b1;

byteReady\_i = 1'b0;

load\_i = 1'b0;

TxByte\_i = 1'b0;

busData\_i = 8'h00;

#10

rst\_i = 1'b0;

#10

byteReady\_i = 1'b1;

#10

busData\_i = 8'h55;

#10

byteReady\_i = 1'b0;

#10

load\_i = 1'b1;

#10

load\_i = 1'b0;

#20

TxByte\_i = 1'b1;

#30

TxByte\_i = 1'b0;

#100

$finish;

end

always #10 clk = ~clk;

endmodule

**Receiver Verilog Code:**

**Module:**

module UARTTx(input clk,input rst,input rcv\_in,output reg [7:0] out,output reg rcv\_valid);

reg [2:0] rCV\_SM\_Main;

reg [2:0] rCV\_SM\_Bit;

reg [3:0] rCV\_Clock\_Count;

reg [7:0] rCV\_Received\_Data;

reg rCV\_start;

parameter INITIAL = 3'd0;parameter START\_BIT = 3'd1;parameter RCV\_DATA = 3'd2;

parameter RCV\_STOP\_BIT = 3'd3;parameter RATE = 115200;parameter CLK\_FRQ = 50000000;parameter BIT\_TIME = CLK\_FRQ / RATE;parameter HALF\_BIT\_TIME = BIT\_TIME / 2;parameter CLKS\_PER\_BIT = BIT\_TIME / 10;

always @(posedge clk) begin

if (rst) begin

rCV\_SM\_Main <= IDLE;

rCV\_SM\_Bit <= 3'd0;

rCV\_Clk\_Count <= 0;

rCV\_Received\_Data <= 8'h00;

rCV\_valid <= 1'b0;

rCV\_start <= 1'b0;

end else begin

case (rCV\_SM\_Main)

IDLE: begin

if (rCV\_in == 1'b0) begin

rCV\_start <= 1'b1;

rCV\_SM\_Main <= START;

rCV\_SM\_Bit <= 3'd0;

rCV\_Received\_Data <= 8'h00;

rCV\_Clk\_Count <= 0;

end

end

START: begin

if (rCV\_start == 1'b0)

begin

rCV\_SM\_Main <= IDLE;

end else if (rCV\_Clk\_Count < HALF\_BIT\_TIME\_1) begin

rCV\_Clk\_Count <= rCV\_Clock\_Count + 1;

end else begin

rCV\_Clock\_Count <= 0;

rCV\_SM\_Main <= Rcv\_DATA;

rCV\_SM\_Bit <= 3'd0;

end

end

Rcv\_DATA: begin

if (rCV\_Clk\_Count < CLKS\_PER\_BIT\_1) begin

rCV\_Clk\_Count <= rCV\_Clock\_Count + 1;

end else begin

rCV\_Clk\_Count <= 0;

rCV\_Received\_Data[r\_SM\_Bit] <= rx\_in;

rCV\_SM\_Bit <= r\_SM\_Bit + 1;

if (rCV\_SM\_Bit == 3'd7) begin

rCV\_SM\_Main <= RCV\_STOP\_BIT;

end

end

end

RCV\_STOP\_BIT: begin

if (rCV\_Clk\_Count < CLKS\_PER\_BIT\_1)

begin

rCV\_Clk\_Count <= rCV\_Clk\_Count + 1;

end else begin

rCV\_Clk\_Count <= 0;

rCV\_valid <= 1'b1;

out <= rCV\_Received\_Data;

rCV\_SM\_Main <= IDLE;

end

end

endcase

if (rCV\_start == 1'b1)

begin

rCV\_start <= 1'b0;

end

end

end

endmodule

**Testbench:**

module TB;

reg clk;

reg rst;

reg rCV\_in;

wire [7:0] out;

wire rCV\_valid;

UARTTx prob8\_receiver(.clk(clk),.rst(rst),.rCV\_in(rCV\_in),.out(out),.rCV\_valid(rCV\_valid));

initial begin

$dumpfile("tb.vcd");

$dumpvars;

clk = 0;

rst = 1;

rCV\_in = 1;

#10

rst = 0;

end

always

#5

clk = ~clk;

initial begin

$dumpfile("tb.vcd");

$dumpvars;

#100

rCV\_in = 0;

#5

rCV\_in = 1;

#5

rCV\_in = 0;

#5

rCV\_in = 1;

#5

rCV\_in = 0;

#5

rCV\_in = 1;

#5

rCV\_in = 0;

#5

rCV\_in = 1;

#5

rCV\_in = 0;

#5

rCV\_in = 1;

#5

rCV\_in = 0;

#5

rCV\_in = 1;

#5

rCV\_in = 0;

#5

rCV\_in = 1;

#5

rCV\_in = 0;

#5

rCV\_in = 1;

repeat(10) begin @(posedge clk) begin

if(rCV\_valid) begin

$display("out = %h, rCV\_valid = %b", out, rCV\_valid);

end

end

end

#100

rCV\_in = 0;

#15

rCV\_in = 1;

#15

rCV\_in = 0;

#15

rCV\_in = 1;

#15

rCV\_in = 0;

#15

rCV\_in = 1;

#15

rCV\_in = 0;

#15

rCV\_in = 1;

#15

rCV\_in = 0;

#15

rCV\_in = 1;

#15

rCV\_in = 0;

#15

rCV\_in = 1;

#15

rCV\_in = 0;

#15

rCV\_in = 1;

#15

rCV\_in = 0;

#15

rCV\_in = 1;

repeat(10) begin

@(posedge clk) begin

if(rCV\_valid) begin

$display("out = %h, rCV\_valid = %b", out,rCV\_valid);

end

end

end

end

endmodule