Name: Simon Gezae

Lab 8 assignment

UART receiver and transmitter RTL design modules and verify them in the testbenches.

Trasmitter

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;

always@(posedge clk)begin

if(rst\_i)begin

curSt\_r <= pIdle;

end

else begin

curSt\_r <= nxtSt\_r ;

end

end

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

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;

end

else if(shift\_r)begin

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

end

end

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

endmodule

module tb;

reg clk, rst\_i, byteReady\_i, load\_i;

reg TxByte\_i;

reg [7:0] busData\_i;

wire serialOut\_o;

UARTTx u1 (clk, rst\_i, byteReady\_i, load\_i, TxByte\_i, busData\_i, serialOut\_o);

always #5 clk = ~clk;

initial begin

{clk, rst\_i, byteReady\_i, load\_i, TxByte\_i} = 0;

$dumpfile("dump.vcd");

$dumpvars;

#2 rst\_i = 1'b1;

#5 rst\_i = 1'b0;

#5 byteReady\_i = 1'b1;

#5 byteReady\_i = 1'b0;

#5 TxByte\_i = 1'b1;

#5 TxByte\_i = 1'b0;

end

endmodule

receiver

module uart\_rx(

input notReady,

input serialIn,

input sampleClk,

input rst,

output reg dataOut,

output reg askReady,

output reg err1,

output reg err2

);

reg [7:0] shiftReg;

reg startBitDetected;

reg stopBitDetected;

reg [2:0] bitCount;

reg [1:0] sampleCount;

parameter BIT\_COUNT\_MAX = 9; // start + 8 data bits + stop

// reset logic

always @(posedge rst) begin

shiftReg <= 0;

startBitDetected <= 0;

stopBitDetected <= 0;

bitCount <= 0;

sampleCount <= 0;

dataOut <= 0;

askReady <= 0;

err1 <= 0;

err2 <= 0;

end

// sampling logic

always @(posedge sampleClk) begin

if (notReady) begin

askReady <= 1;

err1 <= 1;

end else begin

askReady <= 0;

err1 <= 0;

end

if (!startBitDetected && !notReady && !serialIn) begin

startBitDetected <= 1;

bitCount <= 0;

sampleCount <= 0;

end else if (startBitDetected && !stopBitDetected) begin

sampleCount <= sampleCount + 1;

if (sampleCount == 2) begin

shiftReg <= {shiftReg[6:0], serialIn};

bitCount <= bitCount + 1;

sampleCount <= 0;

end

if (bitCount == BIT\_COUNT\_MAX) begin

stopBitDetected <= 1;

startBitDetected <= 0;

dataOut <= shiftReg[7:0];

err2 <= ~serialIn;

end

end else begin

startBitDetected <= 0;

stopBitDetected <= 0;

bitCount <= 0;

sampleCount <= 0;

end

end

endmodule

module uart\_rx\_tb;

reg notReady, serialIn, sampleClk, rst;

wire dataOut, askReady, err1, err2;

uart\_rx dut(

.notReady(notReady),

.serialIn(serialIn),

.sampleClk(sampleClk),

.rst(rst),

.dataOut(dataOut),

.askReady(askReady),

.err1(err1),

.err2(err2)

);

initial begin

notReady = 1;

serialIn = 1;

sampleClk = 0;

rst = 1;

#10 rst = 0;

#10 rst = 1;

// test case 1

#10 notReady = 0;

#10 serialIn = 0;

#10 serialIn = 1;

#10 serialIn = 0;

#10 serialIn = 1;

#10 serialIn = 0;

#10 serialIn = 1;

#10 serialIn = 0;

#10 serialIn = 1;

#10 serialIn = 0;

#10 serial

serialIn = 1;

#10;

$display("test case 1: dataOut=%h, askReady=%d, err1=%d, err2=%d", dataOut, askReady, err1, err2);

if (dataOut !== 8'h55 || askReady !== 0 || err1 !== 0 || err2 !== 0) $error("test case 1 failed");

#10 notReady = 1;

#10 serialIn = 0;

#10 serialIn = 1;

#10 serialIn = 0;

#10 serialIn = 1;

#10 serialIn = 0;

#10 serialIn = 1;

#10 serialIn = 0;

#10 serialIn = 1;

#10 serialIn = 0;

#10 serialIn = 1;

#10;

$display("test case 2: dataOut=%h, askReady=%d, err1=%d, err2=%d", dataOut, askReady, err1, err2);

if (askReady !== 1 || err1 !== 1) $error("test case 2 failed");

// test case 3

#10 notReady = 0;

#10 serialIn = 0; // start bit

#10 serialIn = 1;

#10 serialIn = 0;

#10 serialIn = 1;

#10 serialIn = 0;

#10 serialIn = 1;

#10 serialIn = 0;

#10 serialIn = 1;

#10 serialIn = 0;

#10;

$display("test case 3: dataOut=%h, askReady=%d, err1=%d, err2=%d", dataOut, askReady, err1, err2);

if (err2 !== 1) $error("test case 3 failed");

$display("all test cases passed");

$finish;

end

always #5 sampleClk = ~sampleClk;

endmodule