Ethan Nagelvoort
821234668

Week 5 Lab: UART Rx Lab Part 2

Description:

To complete this lab, I added to my last lab which was the Tx module, which blinks an LED when any of the first eight switches are up. My tx module was explained in the last lab report. To create the Rx module, I have an input called RsRx, a clk input, and an eight bit output called l. I uncommented all the parts needed in the constraint file. I then set 3 registers, one to represent the state of the FSM that will be explained later, one called ll to hold the values of what will be eventually in the l, and num to represent the location of the current switch. I also declare a wire called CLK and use CLK and clk to instantiate clk_div2 which works as a normal clock divider. This clock divider allows me to stay in the same time as the tx module and so the signals from the tx module are sent to the rx module at an appropriate time, not too fast and not too slow. I then go into an always block based on the positive edge of CLK. A case statement is then used to determine which state the code will be in. The code starts at state 2'b00. In this state l is set to ll, which was declared previously as 0. This allows the leds to first be off. Then if RsRx = 0, state becomes 2'b10 and l is also set to 0. This part where l is set to 0, means that every time the transmit process begins, the output leds will turn off. Then in state 2'b10, if num <= 4'b1000 (since there are 8 switches), ll[num] <= RsRx and num increments. This allows register ll to accurately represent the position of the 8 switches. Then in the following else statement, state is set back to 2'b00, num is set back to 0, and l <= ll. Making the l set to ll here allows the eight leds to turn on when the transmission is completed. This ends the case statement. I then have a top module that connects 3 modules, the tx module, rx module, and an LED_controller module. This LED_controller module takes in the outputs of the tx and rx module and accurately sets them to equal to correct led positions in order to turn on or off the leds on the Basys3.

Code:

```
module rx(input RsRx, input clk, output reg [7:0] l);
reg [1:0] state = 2'b00;
reg [7:0] ll = 8'b00000000;
reg [3:0] num = 4'b0000;//current switch
reg [13:0] counter = 32'b00000000000000000000000000000000;
wire CLK;
clk_div2 DUT(.clk(clk), .slowerClk(CLK));
always @(posedge CLK) begin
   case (state)
   2'b00: begin
   l <= ll;
   if(RsRx == 1'b0) begin
```

```
        state <= 2'b10;
        l <= 8'b00000000;
    end
     end
      2'b10: begin
        if (num <= 4'b1000) begin
              ll[num] <= RsRx;
              num <= num + 1;
            end
            else begin
              state <= 2'b00;
              num <= 4'b0000;
              l <= ll;
        end
        end
        endcase
    end

endmodule

module clk_div2 (input clk, output reg slowerClk);
reg [31:0] counter;
always@(posedge clk) begin
if(counter == 100000000) begin//100000000 is threshold, it is close to 1 sec 6250000
slowerClk <= 1;//signal new clk signal
counter <= 0;
end
else begin
slowerClk <= 0;
counter <= counter+1;
end
end

endmodule

module tx(input [15:0] sw, input clk, btnL, output reg RsTx);
parameter state1 = 2'b00;//idle time
parameter state2 = 2'b01;//state that indicates start of transmission
parameter state3 = 2'b10;//state that is longest, it transmitts the data. Uses a counter.
parameter state4 = 2'b11;//state that indicates stop
```

```
wire CLK;
reg [7:0] switch;
reg [1:0] current = 2'b00;
reg [2:0] counter = 3'b000;
reg [1:0] counter2 = 2'b00;
clk_divider  DUT(.clk(clk), .slowerClk(CLK));
always @(posedge CLK) begin
switch = sw[7:0]; //use first 8 switches
case(current)
state1: begin
RsTx <= 1'b1;//idle position, also counts as the second stop bit
if(btnL)begin
current <= state2;
end
end

state2: begin
RsTx <= 1'b0;//indicates the beginning of a transmission
current <= state3;
end

state3: begin
if(switch[counter] == 1'b1)begin
RsTx <= 1'b1;
end
else begin
RsTx <= 1'b0;
end

if(counter<3'b111) begin
counter <= counter + 3'b001;//use counter to go through the switches
end
else begin
counter <= 3'b000;
current = state4;
end
end

state4: begin
RsTx <= 1'b1;//stop bit, the other 1'b01 indicates the second stop bit
```

```
current <= state1;
end


endcase
end
endmodule


module LED_controller(input [7:0] rx_led, input tx_led, clk, output reg [9:0] led);
always @(posedge clk) begin
led [7:0] <= rx_led;
led [8] <= 1'b0;
led [9] <= tx_led;
end
endmodule

module top(input [7:0] sw, input btnL, input clk, output [9:0] led);
wire RsTx;
wire [7:0] l;

tx DUT1 (.sw(sw), .btnL(btnL), .clk(clk), .RsTx(RsTx));

rx DUT2 (.RsRx(RsTx), .clk(clk), .l(l));

LED_controller DUT3 (.rx_led(l), .tx_led(RsTx), .led(led), .clk(clk));

endmodule
```