Ethan Nagelvoort
821234668

Week 4 - UART design and test - Part 1: Tx

<u>Description:</u>

In this lab, I had to transmit an 8 bit signal stored in a register when a push button is pressed by converting that signal into a serial format and shifting the bits asynchronously. The 8 bits of the signal is determined with the first 8 dip switches on the Basys3. I start my state machine module, which does the Uart action, by having a 16 bit sw input, a clk input, a btnL input, and a 2 bit led output. I had to make the led 2 bits and not 1 because I got errors when it was 1 bit, but I leave the second led off entirely in the code and only use the first led. I also uncomment all the necessary properties related to these inputs and outputs. I then set four parameters, each representing a different state that will be used in the state machine. These states are 2 bits and are called the following: state1, state2, state3, and state4. I then declare a wire called CLK, a 8 bit reg called switch, a 2 bit reg called current set to 2'b00, and a 3 bit reg called counter set to 3'b000. I then instantiate a clk divider module like last week's lab and I use clk and CLK to do so. This clk divider module works exactly like the last lab and is the exact same code as last week's lab. We use a clk divider so that the led will flash long enough for people to see it. I then have a always block based on the positive edge of CLK. In it, I first set switch to equal the first 8 bits of sw, since we only need to use the first 8 switches. I then go into a case statement that is based upon current. Since current right now is 2'b00, the program will enter state1 of the case statement. In state1, led is set to 2'b01 which turns the led on. This bit acts as a stop bit in the serial format and is what is used if current is idle and not changing. Then if btnL is pressed, current changes to state2. In state2, led is changed to 2'b00 and this represents the start bit which turns the led off. Current then changes to state3 and in this state the code will cycle through the 8 bits in switch. First off, the counter reg will be used to determine what bit to check in switch. If switch[counter] is 1, the that particular switch is up and if it is 0, then that switch is down. If switch[counter] is 1, then the led will turn on and if not, then the led will turn off. Then after, there is another if-else which checks if counter is above 7. Since we are only checking the first 8 switches, counter cannot go above 7. If it is not above 7, then counter is incremented, and if it isn't, counter is set back to 0 and current is changed to state4. In state4, the led will turn on since a stop bit will occur here and current will change back to state1. Note that the led turning on in state1 represents the second stop bit and its idle position.

<u>Code:</u>

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 02/15/2021 04:49:37 PM
// Design Name:
// Module Name: state_machine
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module state_machine(input [15:0] sw, input clk, btnL, output reg [1:0]led);
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;
clk_divider  DUT(.clk(clk), .slowerClk(CLK));
always @(posedge CLK) begin
switch = sw[7:0]; //use first 8 switches

case(current)
state1: begin
led <= 2'b01;//idle position, also counts as the second stop bit
if(btnL)
current <= state2;
end

state2: begin
led <= 2'b00;//indicates the beginning of a transmission
current <= state3;
end
```

```
state3: begin
if(switch[counter] == 1'b1)
led <= 2'b01;
else
led <= 2'b00;

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

state4: begin
led <= 2'b01;//stop bit, the other 1'b01 also indicates the second stop bit
current <= state1;
end


endcase
end
endmodule

`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 02/15/2021 04:41:57 PM
// Design Name:
// Module Name: clk_divider
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
```

```
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////


module clk_divider (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
slowerClk <= 1;//signal new clk signal
counter <= 0;
end
else begin
slowerClk <= 0;
counter <= counter+1;
end
end



endmodule
```