CS223: Digital Design

Lab#4 Section 1

22202103
Emine İrem Esendemir
12/04/2024

- In this project, we are asked to implement a UART, Universal Asynchronous Receiver-Transmitter. I've defined a top module named UART and it includes 3 sub-module: TXBUF, RXBUF and the segment_display.

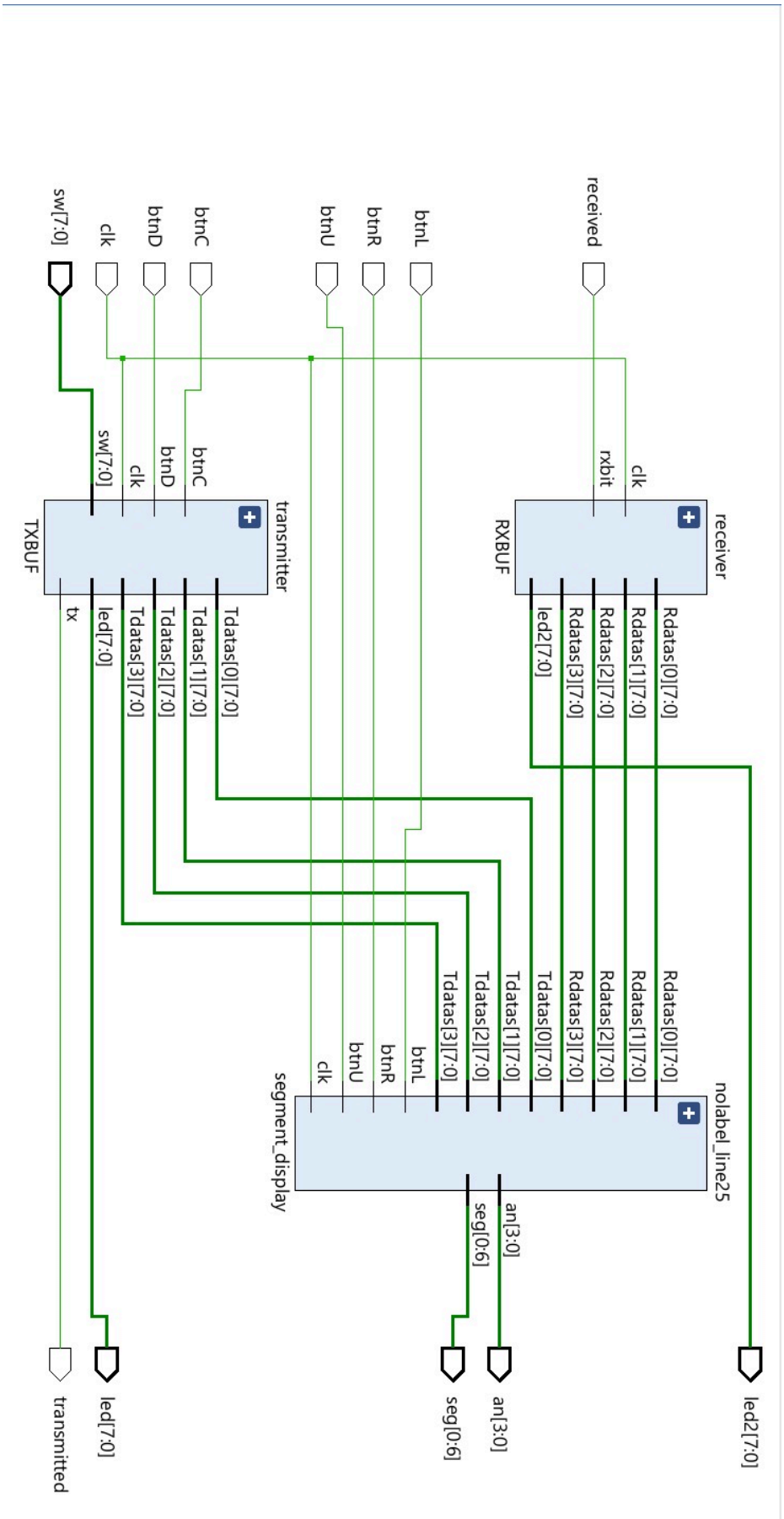
- TXBUF is for transmitting values to the receiver and also it has a 2d array with 4 element which includes the latest 4 element that is loaded. This array utilizes the FIFO principle, the newest loaded element will be loaded to index 0, each middle element will move 1 index, and the last element will be discarded. It has inputs named clock, switches, button D, button C. Clock will be slowed using a clock divider. Switches will be loaded to the array with button D and will be transmitted bit by bit by button C. It has outputs named tx, led, and it also gives the array as the output to UART as it will be used in segment display. Tx is a bit sending the data, which is high in idle state. It lets the receiver know the transmission starts when the TX becomes low, After that start bit, the following datas are inputs. After the inputs, it sends the parity bit to check whether any problem happened in the transmission progress. After that, it sends the stop bit and goes to idle state. The leftmost 8 leds will be open based on the newest loaded date, which is Tdatas[0].

- RXBUF is for receiving the values from the transmitter and also it has a 2d array with 4 element which includes the latest 4 element that is received, tratnsmitted from the other BASYS with button C. This array utilizes the FIFO principle, the newest loaded element will be loaded to index 0, each middle element will move 1 index, and the last element will be discarded. It takes the clk, rxbit as input. Clock will be slowed using a clock divider. Rxbit is the bit transmitted from the other basys. This module has 3 states. It stays in the idle state when the rxbit is one. When the rxbit becomes low, the module goes to receive state and starts the counting the bits and fulls a new array with the values. When the element number is reached, it goes to the shift state. In the shift state, it shifts the elements using the FIFO principle. And puts the newly taken data to Rdatas[0], if the parity pit is 1, otherwise it puts 0 into the array. It assign the Rdatas[3] to led2, the rightmost 8 led in the basys.

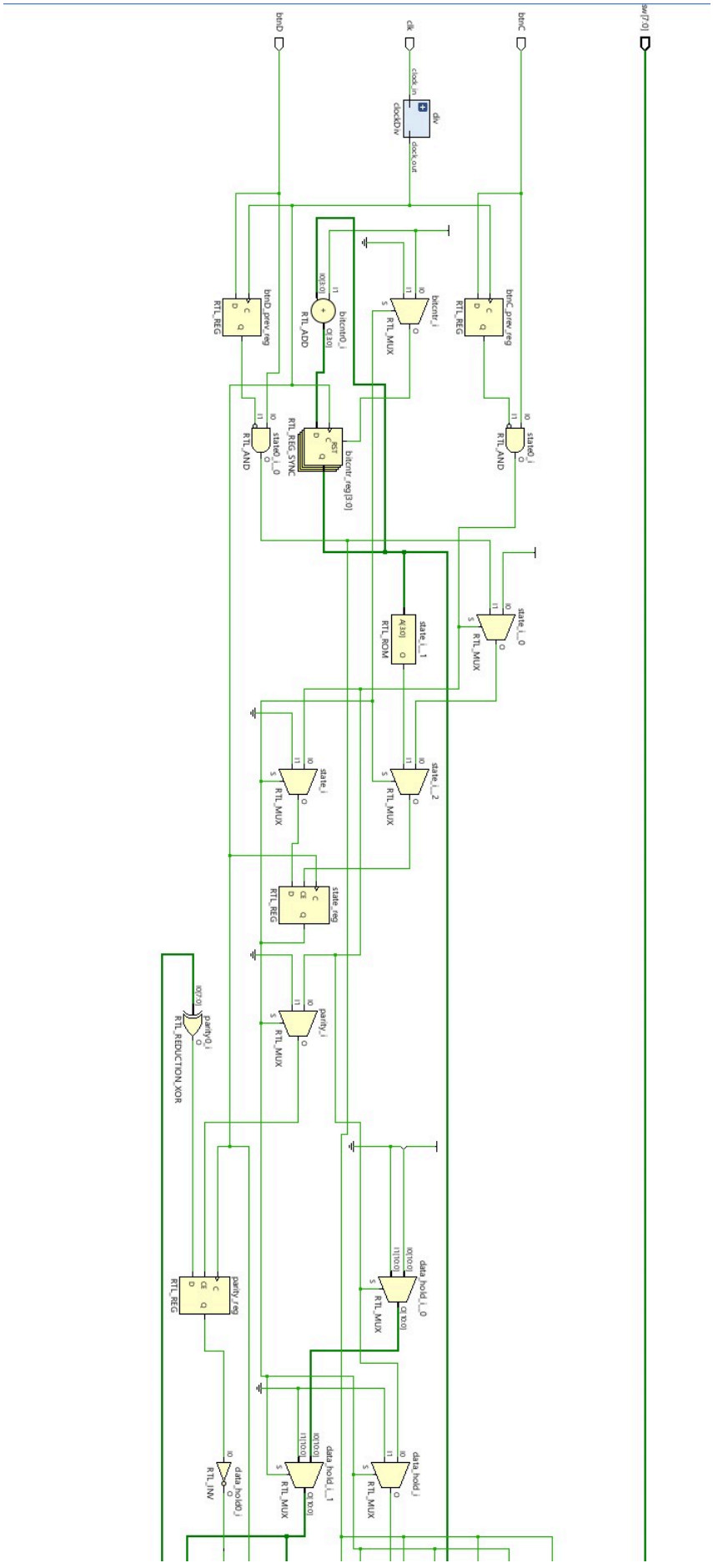
- segment_display uses button debouncer and a helper module which takes the 4 bit value as the inputs and returns how it should be displayed on the segments. It takes 3 button as inputs, for switch between the transmitter and receiver array and for going back and forth in the arrays. It also takes the 2 output arrays of the TXBUF and RXBUF as inputs. It has a boolean for specifying whether the current shown data is transmitted or received ones. It has index for showing the which element is displayed. Also I separated the current data to 2 parts. The rightmost part is turned into the hexadecimal using the helper module and also the leftmost part of the data is shown at the left, which is also turned into the hexadecimal using the helper module.

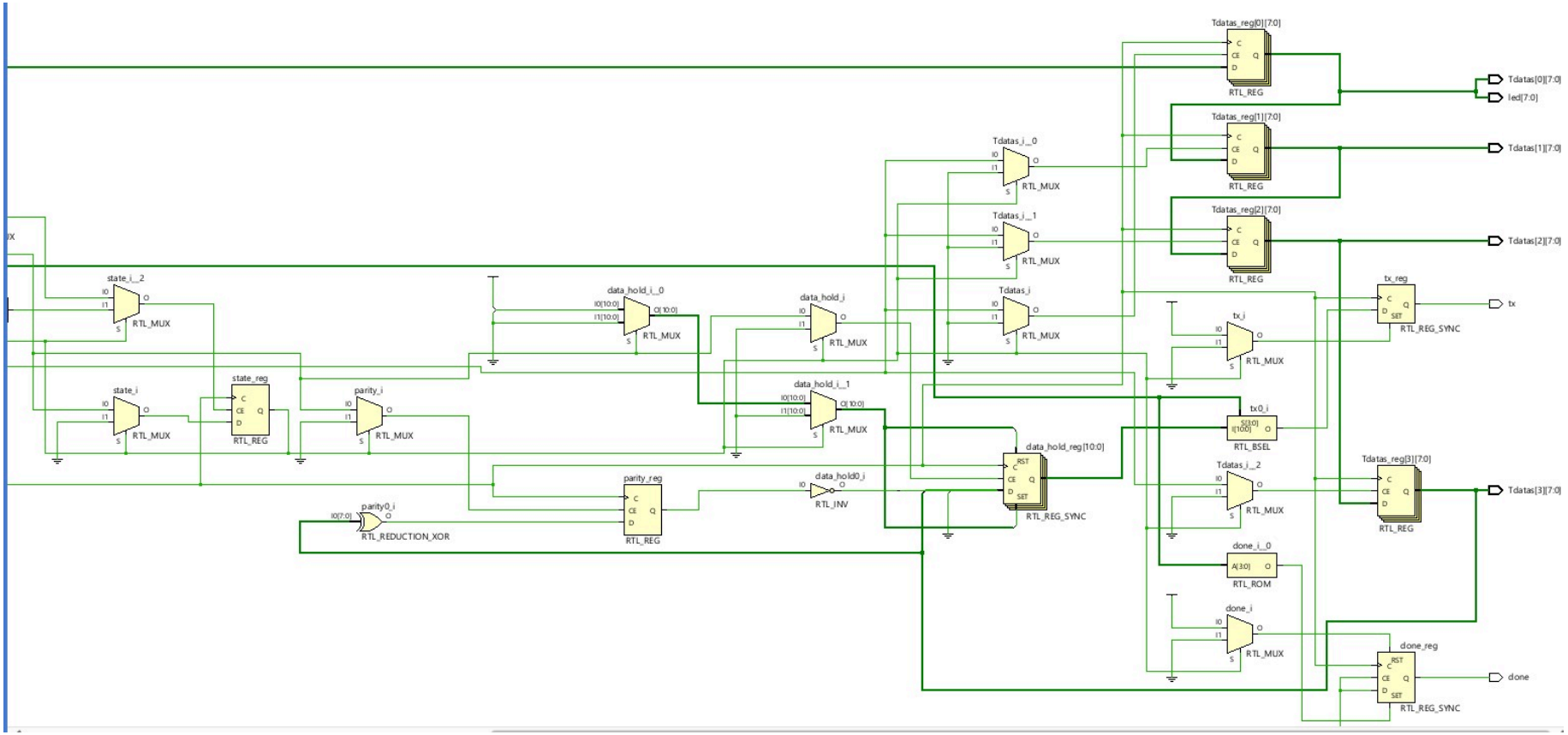
RTL Schematics:

UART:

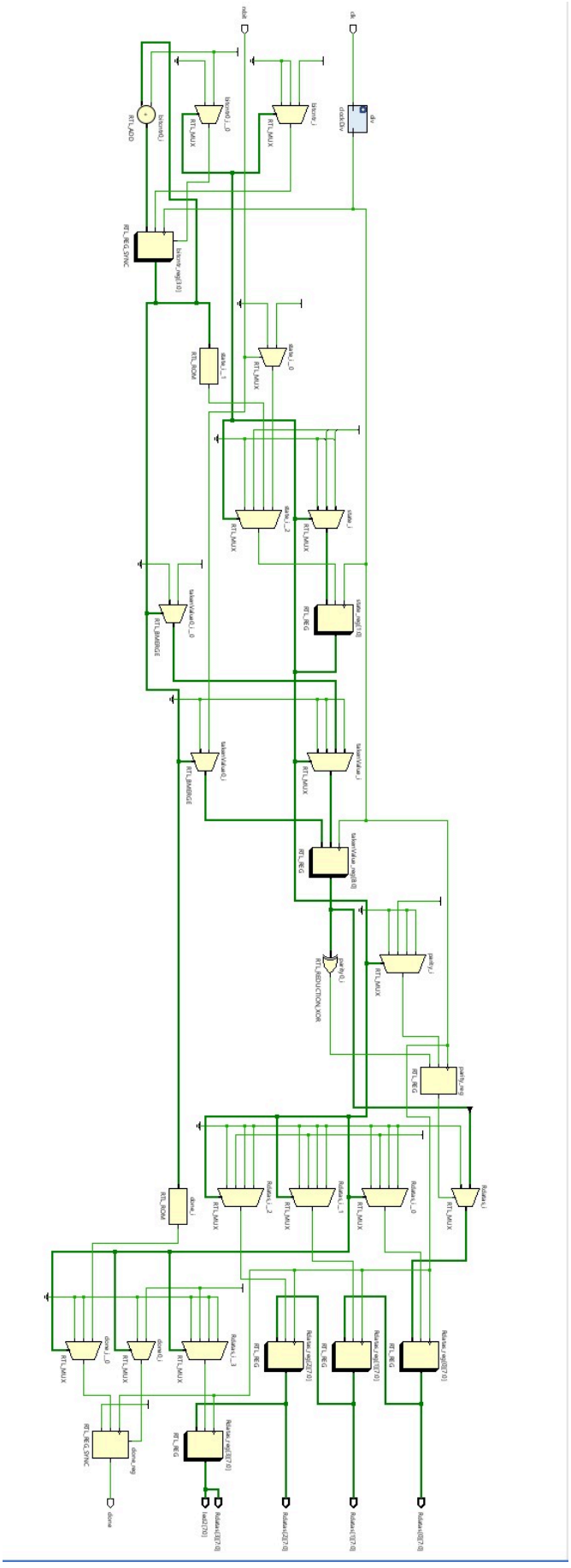


TXBUF:

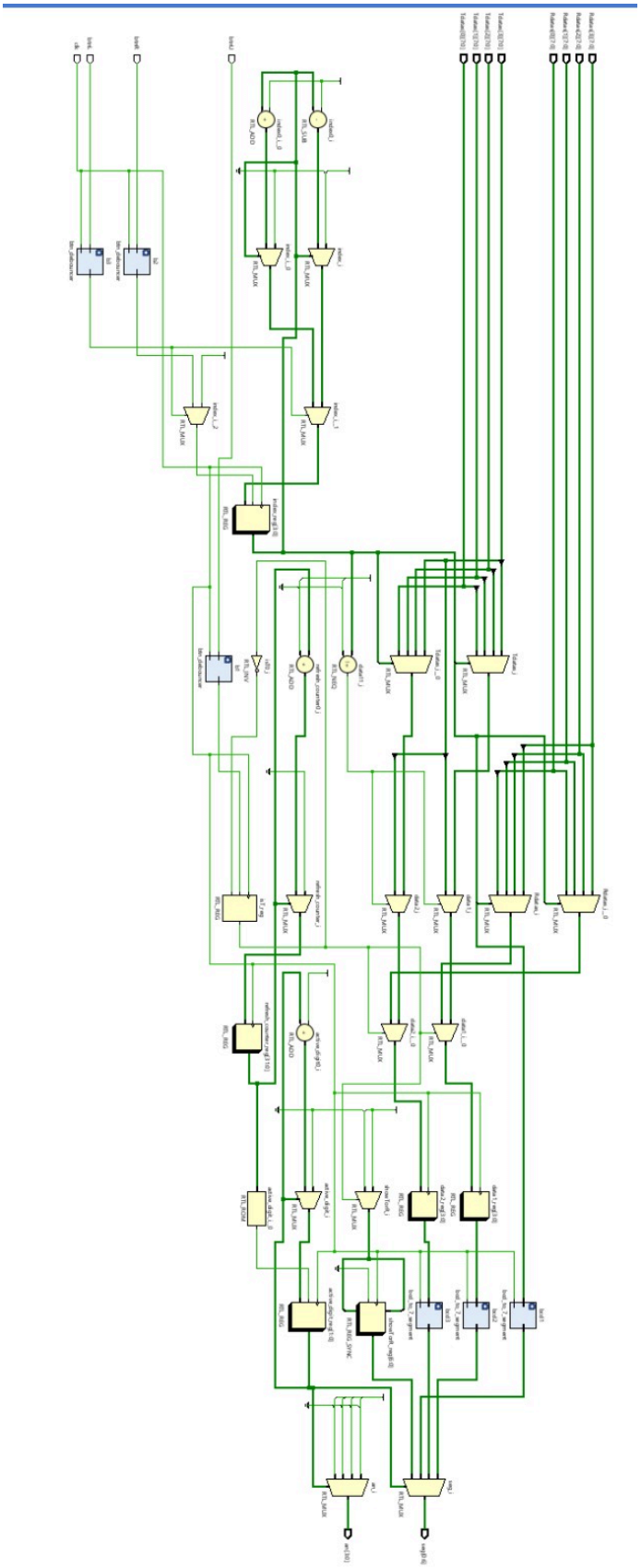

btnD
clk
btnC
sw[7:0]
div
clock_in
clock_out
clockDiv
RTL_REG
btnD_prev_reg
RTL_ADD
bitcnt0_i
RTL_MUX
bitcntr_i
btnC_prev_reg
RTL_AND
state0_i_0
RTL_REG_SYNC
bitcntr_reg[3:0]
state_i_1
RTL_ROM
state_i_0
state_i
state_i_2
state_reg
parity_i
parity0_i
RTL_REDUCTION_XOR
parity_reg
data_hold_i_0
data_hold_i_1
data_hold_i
RTL_INV
data_hold0_i


Tdatas_reg[0][7:0]
RTL_REG
Tdatas[0][7:0]
led[7:0]
Tdatas_reg[1][7:0]
RTL_REG
Tdatas[1][7:0]
Tdatas_reg[2][7:0]
RTL_REG
Tdatas[2][7:0]
Tdatas_i__0
RTL_MUX
Tdatas_i__1
RTL_MUX
Tdatas_i
RTL_MUX
state_i__2
RTL_MUX
data_hold_i__0
RTL_MUX
data_hold_i
RTL_MUX
tx_reg
RTL_REG_SYNC
tx
tx_i
RTL_MUX
state_i
RTL_MUX
state_reg
RTL_REG
parity_i
RTL_MUX
data_hold_i__1
RTL_MUX
tx0_i
RTL_BSEL
data_hold_reg[10:0]
RTL_REG_SYNC
Tdatas_i__2
RTL_MUX
Tdatas_reg[3][7:0]
RTL_REG
Tdatas[3][7:0]
parity_reg
RTL_REG
data_hold0_i
RTL_INV
parity0_i
RTL_REDUCTION_XOR
done_i__0
RTL_ROM
done_i
RTL_MUX
done_reg
RTL_REG_SYNC
done

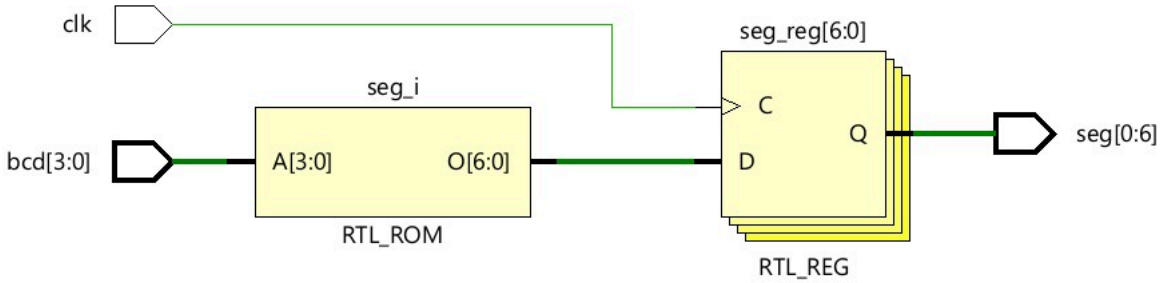
RXBUF:



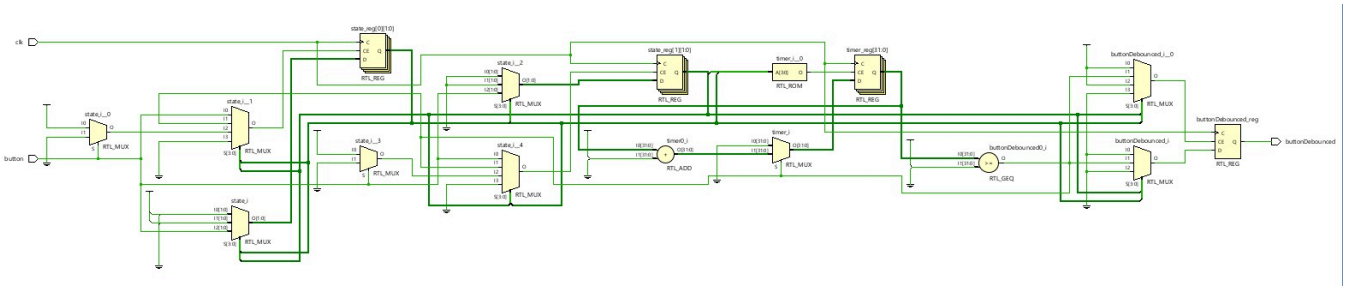
segment_display:



Seven segment converter:

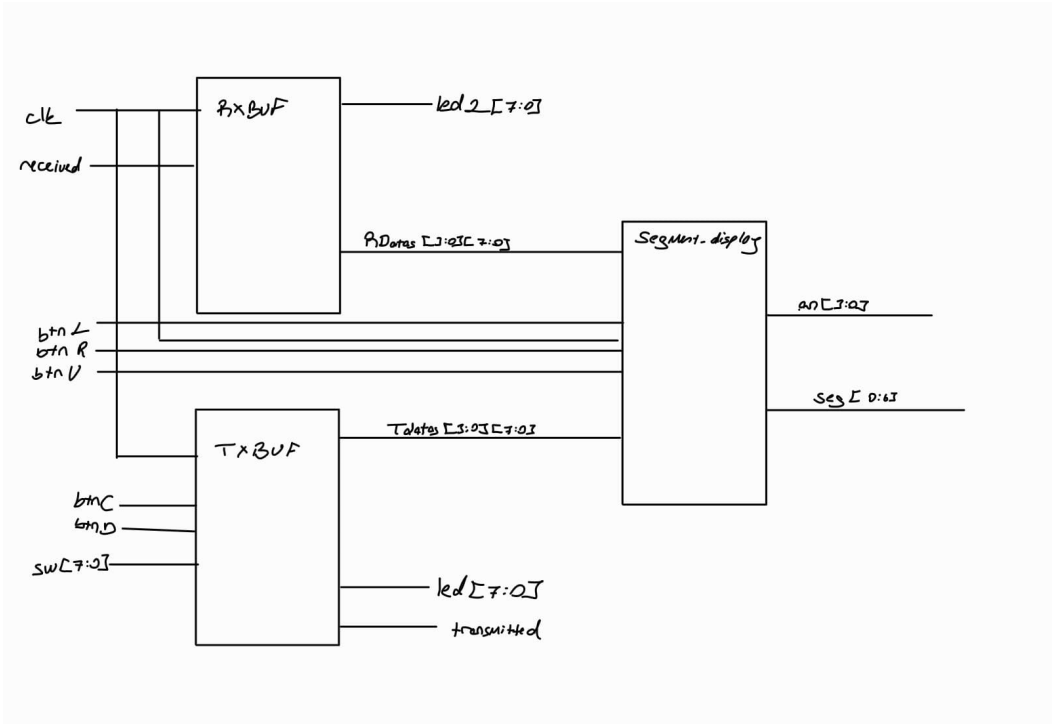


Button Debouncer:



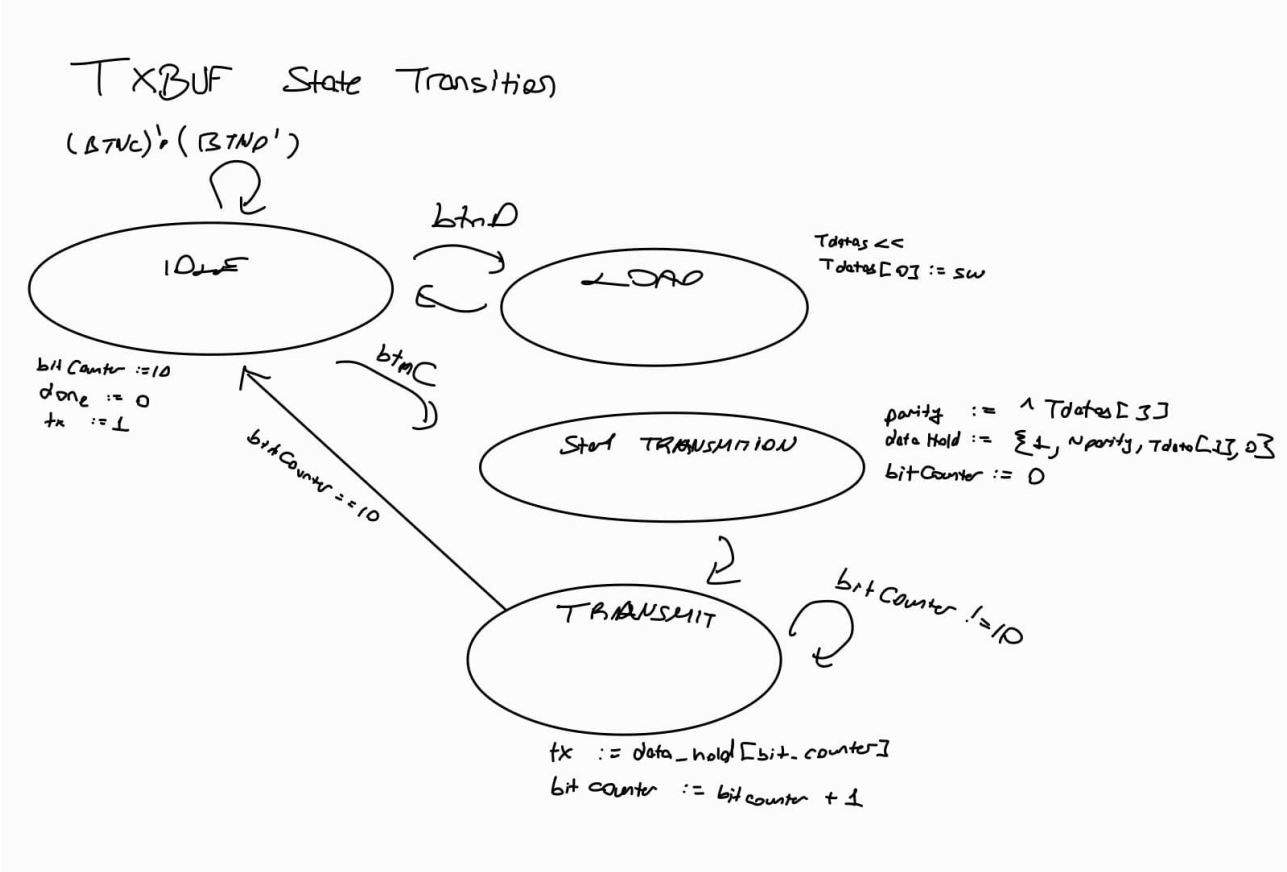
BLOCK DIAGRAM:

UART:

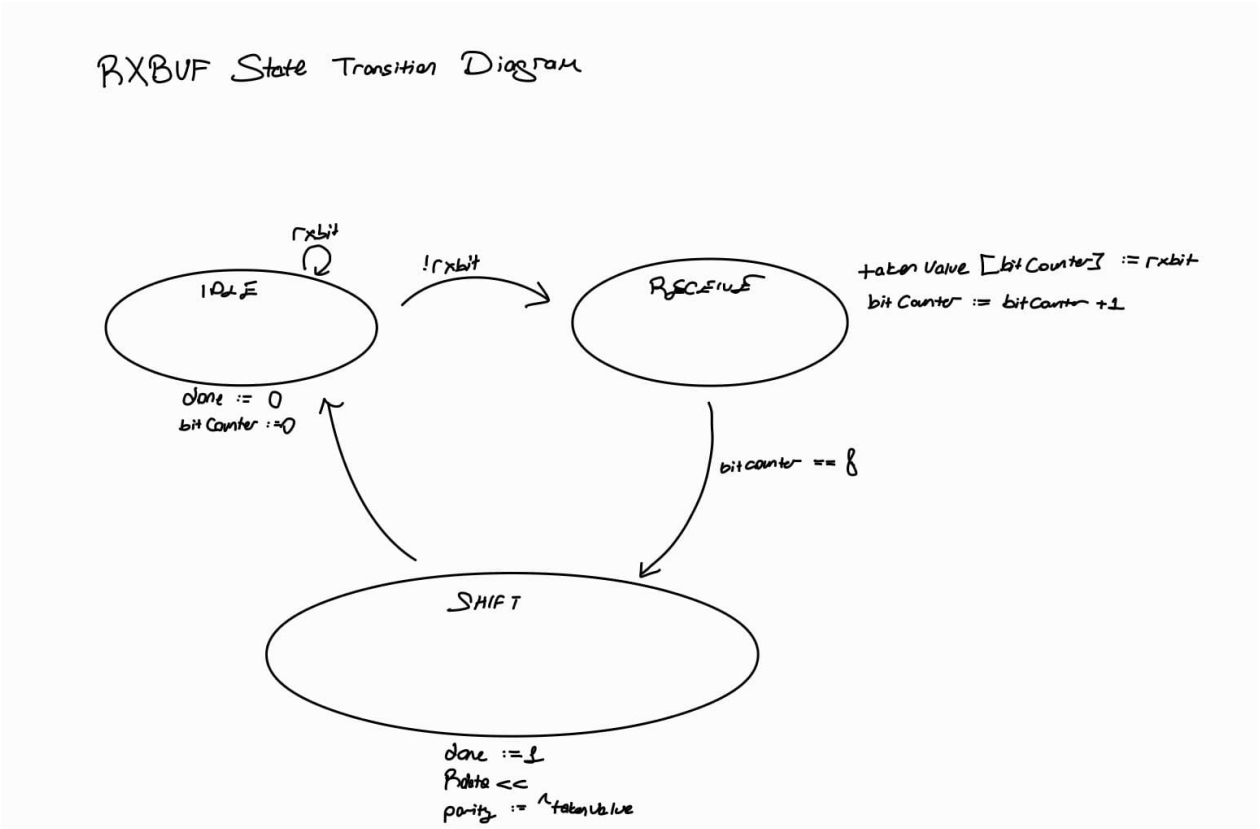


STATE DIAGRAMS:

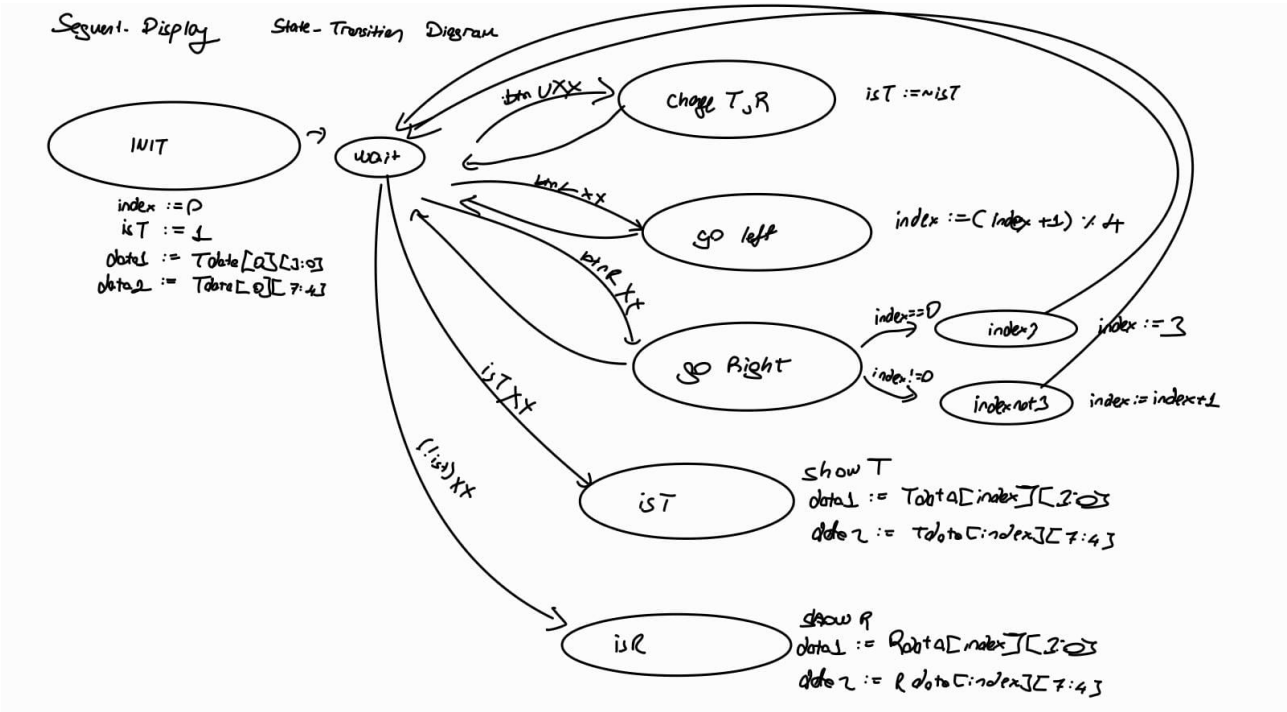
TXBUF:



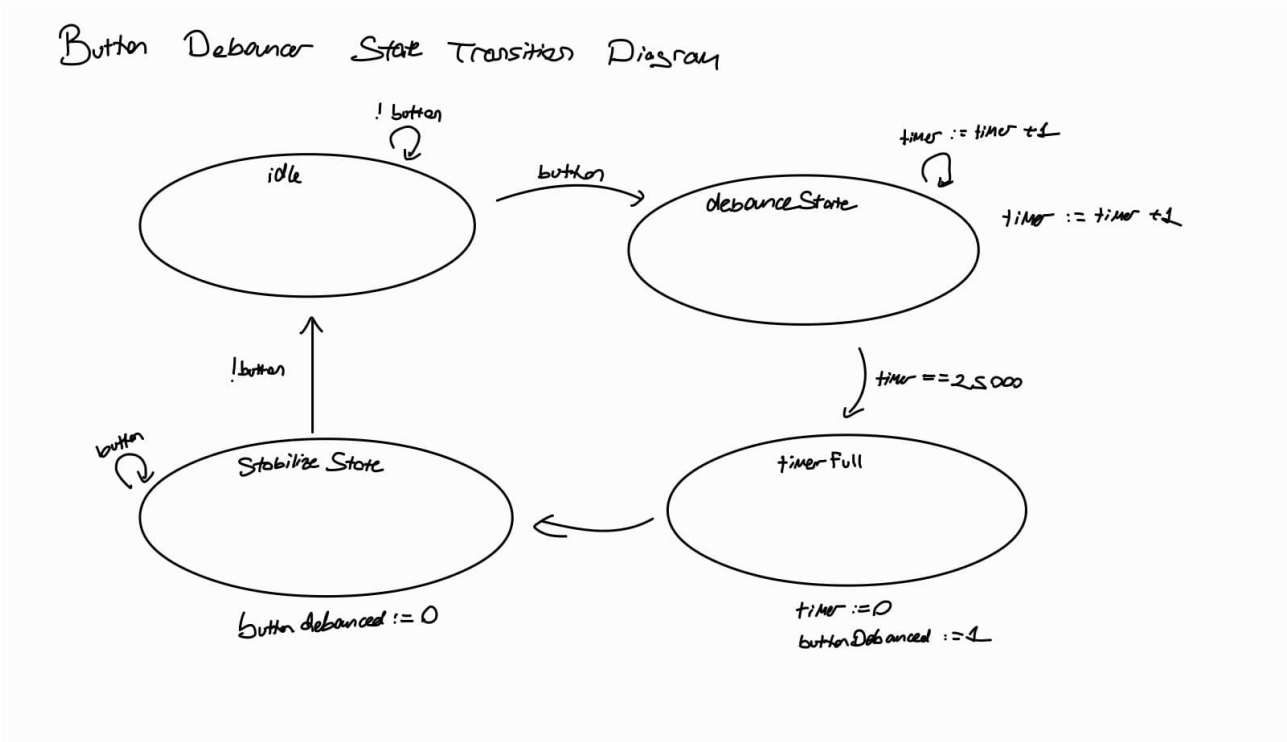
RXBUF:



Segment Display:



Button Debouncer:



APPENDİX:

UART:

```
module UART(
    input logic clk,
    input logic [7:0] sw,
    input logic btnC,
    input logic btnD,
    input logic btnU,
    input logic btnR,
    input logic btnL,
    input logic received,
    output logic transmitted,
    output logic [7:0] led2,
    output logic [7:0] led,
    output logic [0:6] seg,
    output logic [3:0] an
);

   logic Rdone, Tdone;
   logic [3:0][7:0]Rdatas;
   logic [3:0][7:0]Tdatas;

   TXBUF transmitter(clk, sw, btnC, btnD, Tdone, transmitted, led, Tdatas);

   RXBUF receiver(clk, received, Rdone, led2, Rdatas);

   segment_display(clk, btnU, btnR, btnL, Tdatas, Rdatas, an, seg);

endmodule
```

TXBUF:

```
module TXBUF(
        input logic clk,
        input logic [7:0] sw,
        input logic btnC,
        input logic btnD,
        output logic done,
```

```
        output logic tx,
        output logic [7:0] led,
        output logic [3:0][7:0] Tdatas
        );
        typedef enum logic {idle, transmit} typename;
        typename state;
        logic [3:0] bitcntr;
        logic [10:0]data_hold = 0;
        logic parity;
logic btnD_prev;
logic btnC_prev;
initial begin
   Tdatas <= 0;
   btnD_prev <= 0;
   btnC_prev <= 0;
end
clockDiv div(clk,clock_out);

        always_ff @(posedge clock_out) begin
           btnD_prev <= btnD;
           btnC_prev <= btnC;
        case (state)
                 idle: begin
                    bitcntr <=0;
                 done <= 1'b0;
                 tx <= 1'b1;
                 if(btnD && ~btnD_prev) begin
                Tdatas [3:1] <= Tdatas [2:0];
                Tdatas[0] <= sw;
                state <= idle;
                 end
                 if(btnC && ~btnC_prev) begin
                    parity <= ^Tdatas[3];
                    data_hold <= {1'b1, ~parity, Tdatas[3], 1'b0};
                 state <= transmit;
                 bitcntr <=0;
                 end
```

```
                end
                transmit: begin
            tx <= data_hold[bitcntr];
            bitcntr<=bitcntr+1;
            if(bitcntr==10) begin
                done <= 1'b1;
                state <= idle;
            end
                end
        endcase
        end
        assign led = Tdatas[0];
endmodule
```

RXBUF:

```
module RXBUF(
        input logic clk,
        input logic rxbit,
        output logic done,
        output logic [7:0] led2,
        output logic [3:0][7:0] Rdatas
);
        typedef enum logic [1:0]{idle, receive, shift} tstate;
        tstate state;
        logic [3:0] bitcntr;
        logic clock_out;
        logic parity = 0;
        logic [8:0] takenValue;
   initial begin
      Rdatas <= 0;
   end
        clockDiv div(clk,clock_out);
        always_ff @(posedge clock_out) begin
        case (state)
                idle: begin
                done <= 1'b0;
```

```
                bitcntr <=0;
                if(!rxbit) begin
                state<=receive;
                end
                end
                receive: begin
            takenValue[bitcntr] <= rxbit;
            bitcntr<=bitcntr+1;
            if(bitcntr==8) begin
                done <= 1'b1;
                state <= shift;
            end
                end
                shift: begin
                    Rdatas[3:1] <= Rdatas[2:0];
            parity <= ^takenValue;
            if(parity) Rdatas[0] <= takenValue[7:0];
            else Rdatas[0] <= 0;
            state <= idle;
                end
        endcase
        end
        assign led2 = Rdatas[3];
endmodule
```

segment_display:

```
module segment_display(
    input logic clk,
    input logic btnU,
    input logic btnR,
    input logic btnL,
    input logic [3:0][7:0] Tdatas,
    input logic [3:0][7:0] Rdatas,
    output logic [3:0]an,
    output logic [0:6]seg
    );
```

```
logic [0:6] showTorR;
logic [0:6] showIndex;
logic [3:0] data1;
logic [3:0] data2;
logic [0:6] showData1;
logic [0:6] showData2;
logic [3:0] index;
logic isT;
logic btnU_db;
logic btnR_db;
logic btnL_db;

initial begin
    index <= 0;
    isT <= 1;
    data1[3:0] <= Tdatas[0][3:0];
    data2[3:0] <= Tdatas[0][7:4];
    btnU_db <= 0;
end

reg [1:0] active_digit = 0;
integer refresh_counter = 0;
always @(posedge clk) begin
    refresh_counter <= refresh_counter + 1;
    if (refresh_counter == 250000) begin
        refresh_counter <= 0;
        active_digit <= active_digit + 1;
        if (active_digit == 3) active_digit <= 0;
    end
end
always @(*) begin
    case (active_digit)
        2'b00: begin
            seg = showData1;
            an = 4'b1110;
        end
        2'b01: begin
```

```
            seg = showData2;
            an = 4'b1101;
        end
        2'b10: begin
            seg = showIndex;
            an = 4'b1011;
        end
        2'b11: begin
            seg = showTorR;
            an = 4'b0111;
        end
    endcase
end

always_ff@(posedge clk) begin
    if(btnU_db) begin
        isT <= ~isT;
    end
     if (btnR_db) begin
        if (index == 3)
            index <= 0;
        else
            index <= index + 1;
    end
    if (btnL_db) begin
        if (index == 0)
            index <= 3;
        else
        index <= index - 1;
    end
    if(isT)begin
    if(index != 3) begin
        showTorR <=  7'b1110000;
        data1 <= Tdatas[index][3:0];
        data2 <= Tdatas[index][7:4];
    end else begin
        showTorR <=  7'b1110000;
```

```
            data1 <= Tdatas[3][3:0];
            data2 <= Tdatas[3][7:4];
        end
        end
        else begin
            showTorR <= 7'b1111010;
            data1 <= Rdatas[index][3:0];
            data2 <= Rdatas[index][7:4];
        end

    end
    btn_debouncer b1(clk, btnU, btnU_db);
    btn_debouncer b2(clk, btnR, btnR_db);
    btn_debouncer b3(clk, btnL, btnL_db);
    bcd_to_7_segment bcd1(clk, index, showIndex);
    bcd_to_7_segment bcd2(clk, data1, showData1);
    bcd_to_7_segment bcd3(clk, data2, showData2);
endmodule
```

clockDiv:

```
`timescale 1ns / 1ps
module clockDiv(input logic clock_in,output logic clock_out
    );
    parameter baud_rate = 9600;
    logic[27:0] counter;
    always @(posedge clock_in)
    begin
        counter <= counter + 1;
        if(counter>=(100_000_000)/baud_rate)
        begin
            counter <= 0;
            clock_out <= ~clock_out;
        end
    end
endmodule
```

btn_debouncer:

```
`timescale 1ns / 1ps
    module btn_debouncer(
        input logic clk, button,
        output logic buttonDebounced
    );
    integer timer = 0;
    typedef enum logic [1:0] {idleState, debounceState, stabilizeState} stateType;
    stateType [1:0] state = idleState;
    always_ff @(posedge clk) begin
        case (state)
            idleState: begin
                buttonDebounced <= 0;
                if(button) begin
                    state <= debounceState;
                end
            end
            debounceState: begin
                timer <= timer+1;
                if(timer >= 25000)begin
                    timer <= 0;
                    buttonDebounced <= 1;
                    state <= stabilizeState;
                end
            end
            stabilizeState: begin
                buttonDebounced <= 0;
                if(~button) begin
                    state <= idleState;
                end
            end
        endcase
    end
endmodule
```

bcd_to_7_segment:

```
module bcd_to_7_segment(
    input logic clk,
    input logic [3:0] bcd,
    output logic [0:6] seg  // segments a-g
);
    always @(posedge clk) begin
        case(bcd)
            4'b0000: seg = 7'b0000001; // 0
            4'b0001: seg = 7'b1001111; // 1
            4'b0010: seg = 7'b0010010; // 2
            4'b0011: seg = 7'b0000110; // 3
            4'b0100: seg = 7'b1001100; // 4
            4'b0101: seg = 7'b0100100; // 5
            4'b0110: seg = 7'b0100000; // 6
            4'b0111: seg = 7'b0001111; // 7
            4'b1000: seg = 7'b0000000; // 8
            4'b1001: seg = 7'b0000100; // 9
            4'b1010: seg = 7'b0001000; // A
            4'b1011: seg = 7'b1100000; // B
            4'b1100: seg = 7'b0110001; // C
            4'b1101: seg = 7'b1000010; // D
            4'b1110: seg = 7'b0110000; // E
            4'b1111: seg = 7'b0111000; // F
            default: seg = 7'b1111111;
        endcase
    end
endmodule
```

CONSTRAINTS:

```
# Clock signal
set_property PACKAGE_PIN W5 [get_ports clk]
        set_property IOSTANDARD LVCMOS33 [get_ports clk]
# create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]

# Switches
```

```
set_property PACKAGE_PIN V17 [get_ports {sw[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {sw[0]}]
set_property PACKAGE_PIN V16 [get_ports {sw[1]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {sw[1]}]
set_property PACKAGE_PIN W16 [get_ports {sw[2]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {sw[2]}]
set_property PACKAGE_PIN W17 [get_ports {sw[3]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {sw[3]}]
set_property PACKAGE_PIN W15 [get_ports {sw[4]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {sw[4]}]
set_property PACKAGE_PIN V15 [get_ports {sw[5]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {sw[5]}]
set_property PACKAGE_PIN W14 [get_ports {sw[6]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {sw[6]}]
set_property PACKAGE_PIN W13 [get_ports {sw[7]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {sw[7]}]
set_property PACKAGE_PIN V2 [get_ports {sw[8]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {sw[8]}]
set_property PACKAGE_PIN T3 [get_ports {sw[9]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {sw[9]}]
set_property PACKAGE_PIN T2 [get_ports {sw[10]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {sw[10]}]
set_property PACKAGE_PIN R3 [get_ports {sw[11]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {sw[11]}]
set_property PACKAGE_PIN W2 [get_ports {sw[12]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {sw[12]}]
set_property PACKAGE_PIN U1 [get_ports {sw[13]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {sw[13]}]
set_property PACKAGE_PIN T1 [get_ports {sw[14]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {sw[14]}]
set_property PACKAGE_PIN R2 [get_ports {sw[15]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {sw[15]}]

# LEDs
set_property PACKAGE_PIN U16 [get_ports {led[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {led[0]}]
set_property PACKAGE_PIN E19 [get_ports {led[1]}]
```

```
        set_property IOSTANDARD LVCMOS33 [get_ports {led[1]}]
set_property PACKAGE_PIN U19 [get_ports {led[2]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {led[2]}]
set_property PACKAGE_PIN V19 [get_ports {led[3]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {led[3]}]
set_property PACKAGE_PIN W18 [get_ports {led[4]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {led[4]}]
set_property PACKAGE_PIN U15 [get_ports {led[5]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {led[5]}]
set_property PACKAGE_PIN U14 [get_ports {led[6]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {led[6]}]
set_property PACKAGE_PIN V14 [get_ports {led[7]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {led[7]}]
set_property PACKAGE_PIN V13 [get_ports {led2[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {led2[0]}]
set_property PACKAGE_PIN V3 [get_ports {led2[1]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {led2[1]}]
set_property PACKAGE_PIN W3 [get_ports {led2[2]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {led2[2]}]
set_property PACKAGE_PIN U3 [get_ports {led2[3]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {led2[3]}]
set_property PACKAGE_PIN P3 [get_ports {led2[4]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {led2[4]}]
set_property PACKAGE_PIN N3 [get_ports {led2[5]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {led2[5]}]
set_property PACKAGE_PIN P1 [get_ports {led2[6]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {led2[6]}]
set_property PACKAGE_PIN L1 [get_ports {led2[7]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {led2[7]}]


#7 segment display
set_property PACKAGE_PIN W7 [get_ports {seg[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {seg[0]}]
set_property PACKAGE_PIN W6 [get_ports {seg[1]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {seg[1]}]
set_property PACKAGE_PIN U8 [get_ports {seg[2]}]
```

```
        set_property IOSTANDARD LVCMOS33 [get_ports {seg[2]}]
set_property PACKAGE_PIN V8 [get_ports {seg[3]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {seg[3]}]
set_property PACKAGE_PIN U5 [get_ports {seg[4]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {seg[4]}]
set_property PACKAGE_PIN V5 [get_ports {seg[5]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {seg[5]}]
set_property PACKAGE_PIN U7 [get_ports {seg[6]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {seg[6]}]
set_property PACKAGE_PIN V7 [get_ports dp]
        set_property IOSTANDARD LVCMOS33 [get_ports dp]
set_property PACKAGE_PIN U2 [get_ports {an[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {an[0]}]
set_property PACKAGE_PIN U4 [get_ports {an[1]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {an[1]}]
set_property PACKAGE_PIN V4 [get_ports {an[2]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {an[2]}]
set_property PACKAGE_PIN W4 [get_ports {an[3]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {an[3]}]

#Buttons
set_property PACKAGE_PIN U18 [get_ports btnC]
        set_property IOSTANDARD LVCMOS33 [get_ports btnC]
set_property PACKAGE_PIN T18 [get_ports btnU]
        set_property IOSTANDARD LVCMOS33 [get_ports btnU]
set_property PACKAGE_PIN W19 [get_ports btnL]
        set_property IOSTANDARD LVCMOS33 [get_ports btnL]
set_property PACKAGE_PIN T17 [get_ports btnR]
        set_property IOSTANDARD LVCMOS33 [get_ports btnR]
set_property PACKAGE_PIN U17 [get_ports btnD]
        set_property IOSTANDARD LVCMOS33 [get_ports btnD]

##Pmod Header JC
##Sch name = JC1
set_property PACKAGE_PIN K17 [get_ports {transmitted}]
        set_property IOSTANDARD LVCMOS33 [get_ports {transmitted}]
##Sch name = JC2
```

```
set_property PACKAGE_PIN M18 [get_ports {received}]
        set_property IOSTANDARD LVCMOS33 [get_ports {received}]
```