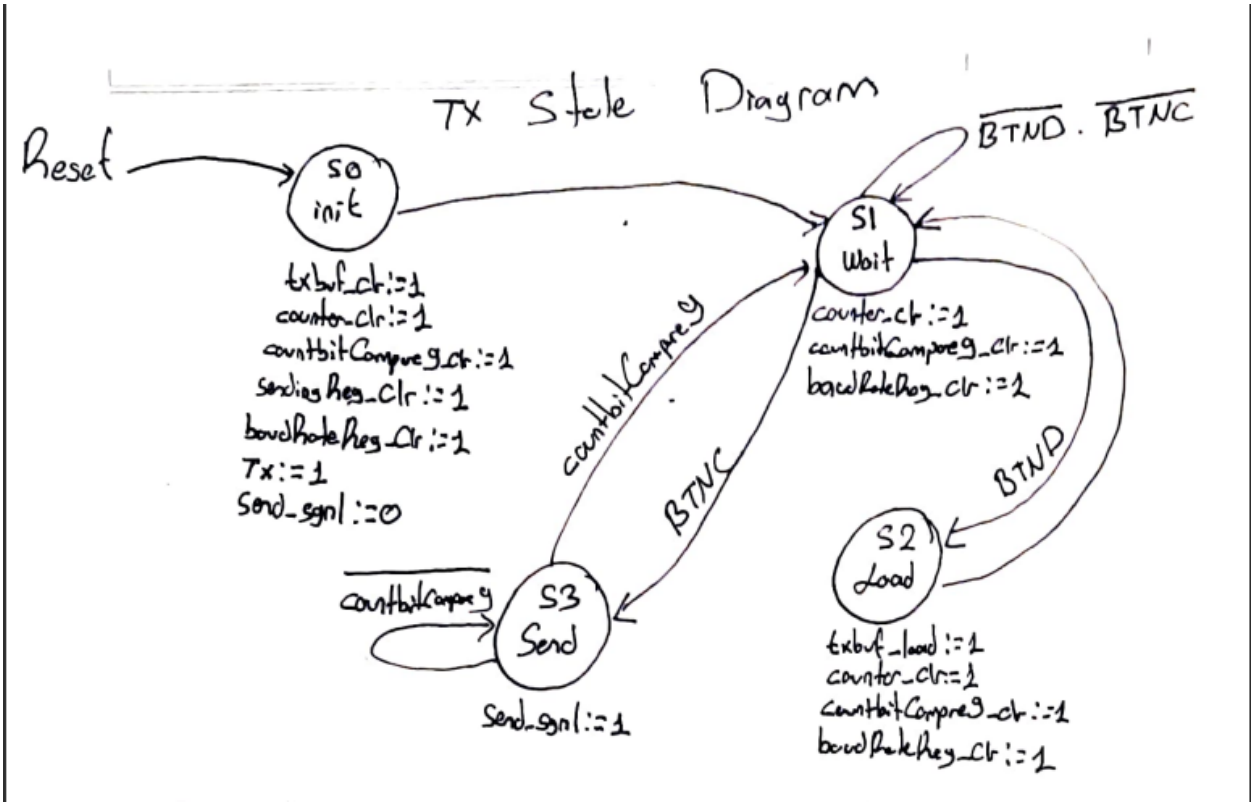
Explanation of Implementation

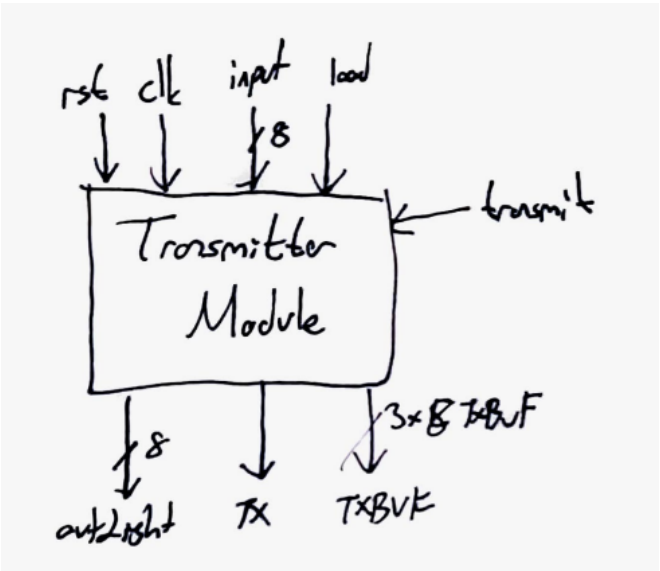
The transmitting module has been implemented through 1 8-bit, four depth array TXBUF that is used with bitwise shift right operation to create the shift register. Another temporary reg sendingReg is used to send the chosen byte without altering the original value; thus, protecting the data from serialization for TX signal. TX signal is starts as high and only dropped when the transmission starts as a starting bit, in every state change except for transmission, the sendingReg , start and stop bits are updated through the TXBUF to keep the new value in the sendingReg, read to be transmitted. Furthermore, the baud rate is chosen as 115200 and 1 bit is sent throughout the 867 clock cycles. Transmission starts when dataTransmit is high, which starts both the baudratecounter and bitcounter. Bit counter is used to count the number of bits sent over TX throughout the baudrate while baudratecounter is used to keep the current data across the 867 clock cycles. After the bitCounter is 10, the transmission stops, TX becomes 1 and it goes back t o waiting state for further input. sendingReg is serialized through bitwise shift right operator and [] operator to extract the one bit into TX signal.

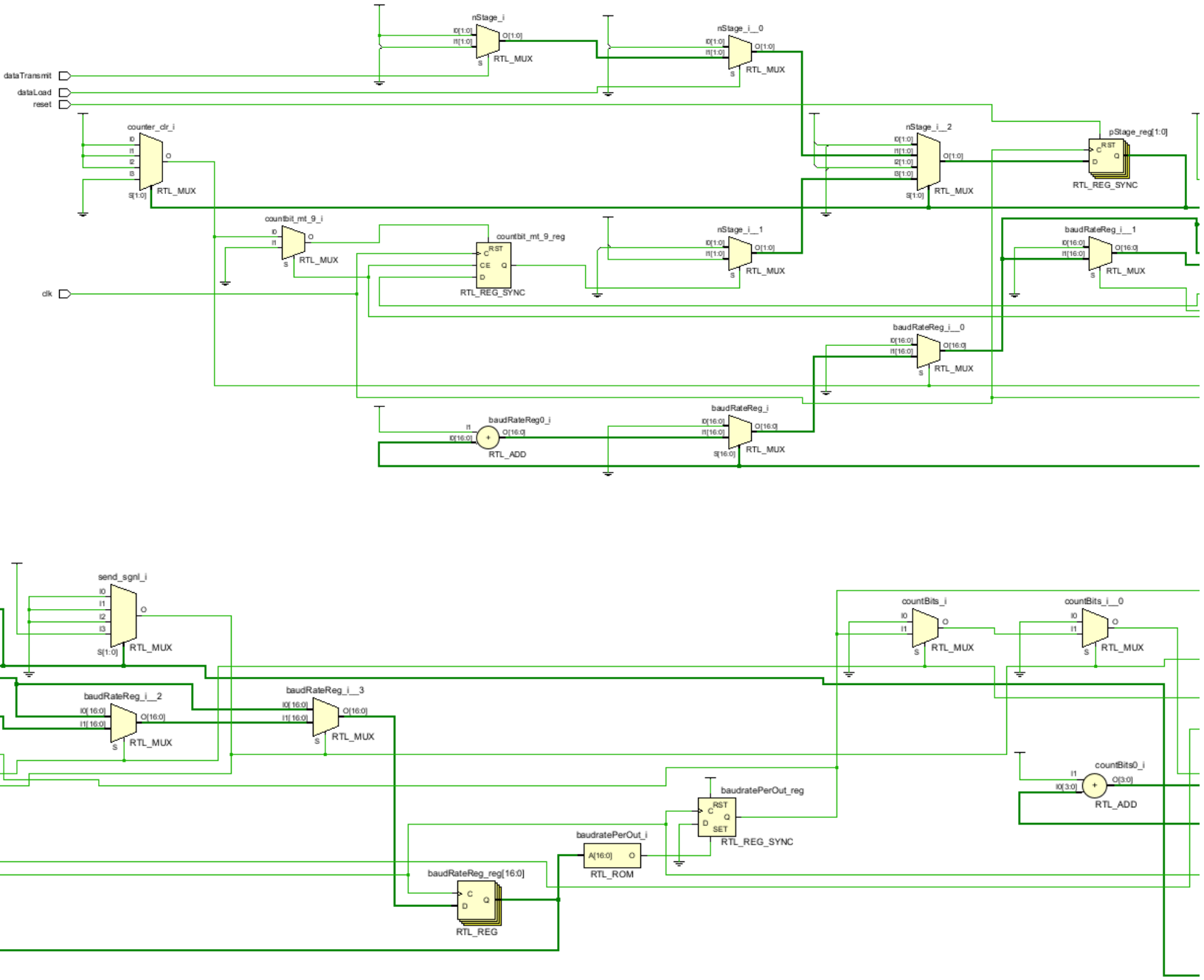
The receiver module has a similar implementation with 4 x 8-bit RXBUF that is used through bitwise left shift operation to transfer the new data throughout the register. Recieveing_sgnl is used to detect the change of RX from 1 to 0, which starts the transition from waiting to receiving and sampling the RX signal. RX signal is sampled by counting half of the 867 clock which is 433 to midSample the current RX to the place in the reg which is the value of countBit and when the full baudrate is complete, it samples the next data in sync with TX while also similar to transmission module, counting the data received and waiting for stop bit to process the information. When TX = 1 from 0 and bitCount is 10 , the module goes into load state to load the unserialized RX input into the RXBUF while shifting all the other concurrent data to left.

The seven-segment display takes the input of transmission and receiver modules, which outputs their concurrent RXBUF AND TXBUF. According to the input, the module loads the chosen data from these two registers at load state and shows it through sevenSegmentOut reg. It is primarily same as state diagram.

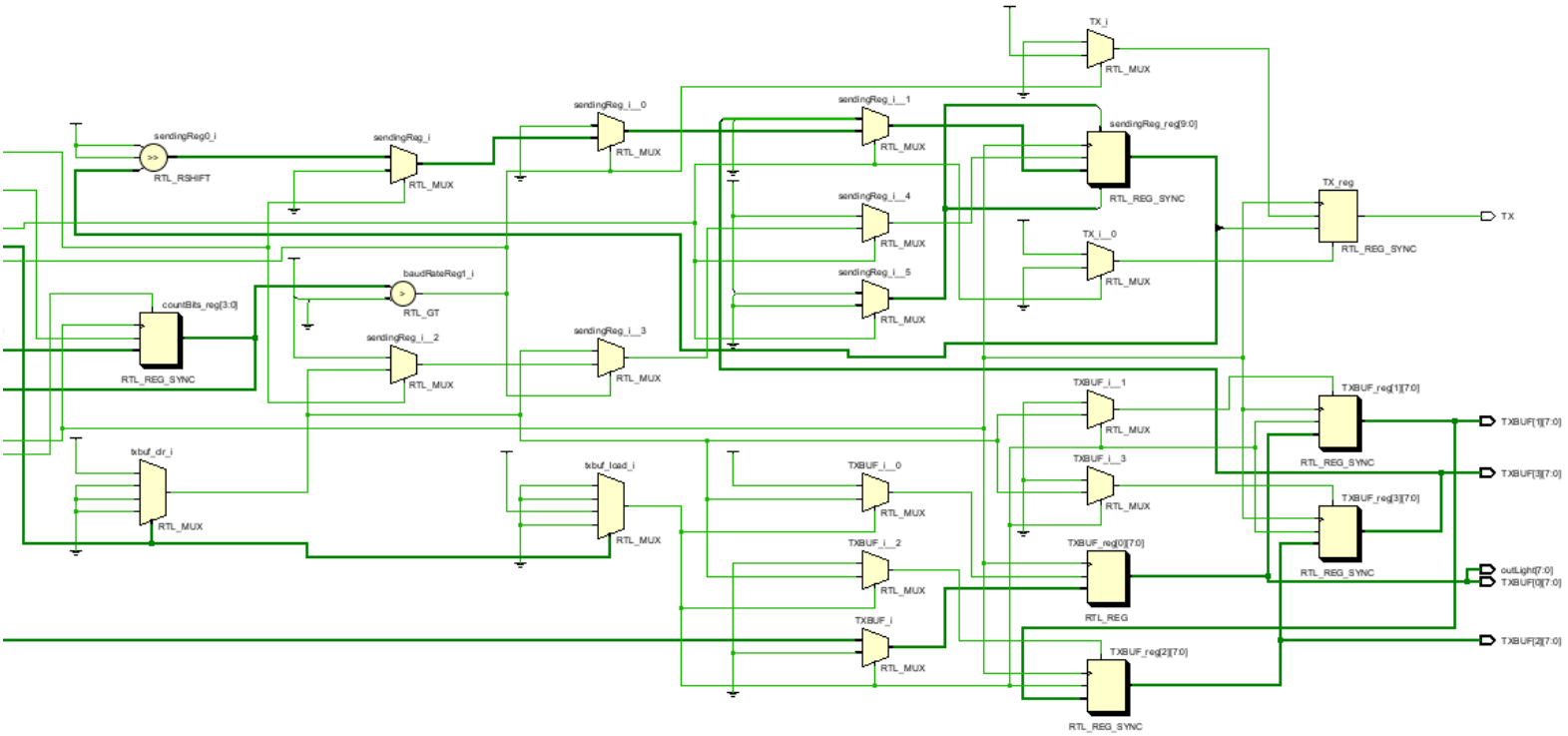
Main module connects all 3 essential modules for the operation of UART.

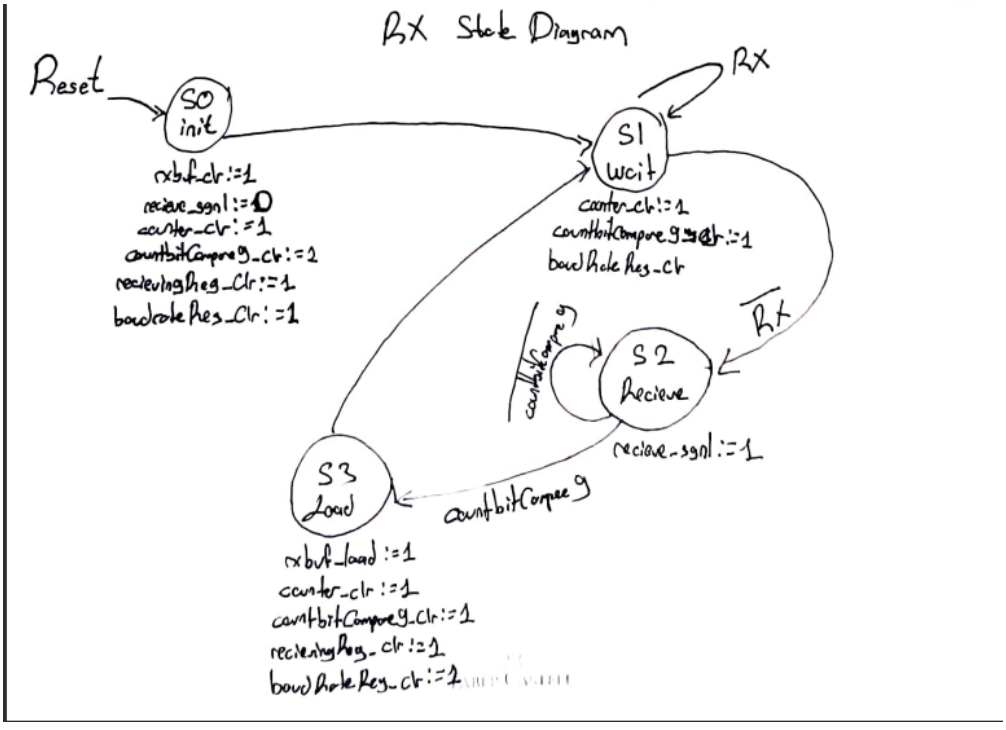

TX State Diagram
Reset
S0
init
txbuf_clr := 1
counter_clr := 1
countbitCompare9_clr := 1
sendingReg_clr := 1
baudRateReg_clr := 1
Tx := 1
Send_sgnl := 0
S1
Wait
BTND . BTNC
counter_clr := 1
countbitCompare9_clr := 1
baudRateReg_clr := 1
countbitCompare9
BTNC
BTND
S2
Load
txbuf_load := 1
counter_clr := 1
countbitCompare9_clr := 1
baudRateReg_clr := 1
countbitCompare9
S3
Send
Send_sgnl := 1

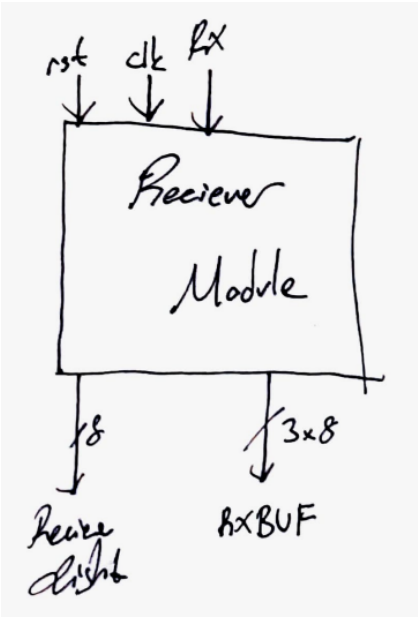

rst
clk
input
load
8
transmit
Transmitter
Module
8
3x8 TXBUF
TX
TXBUF

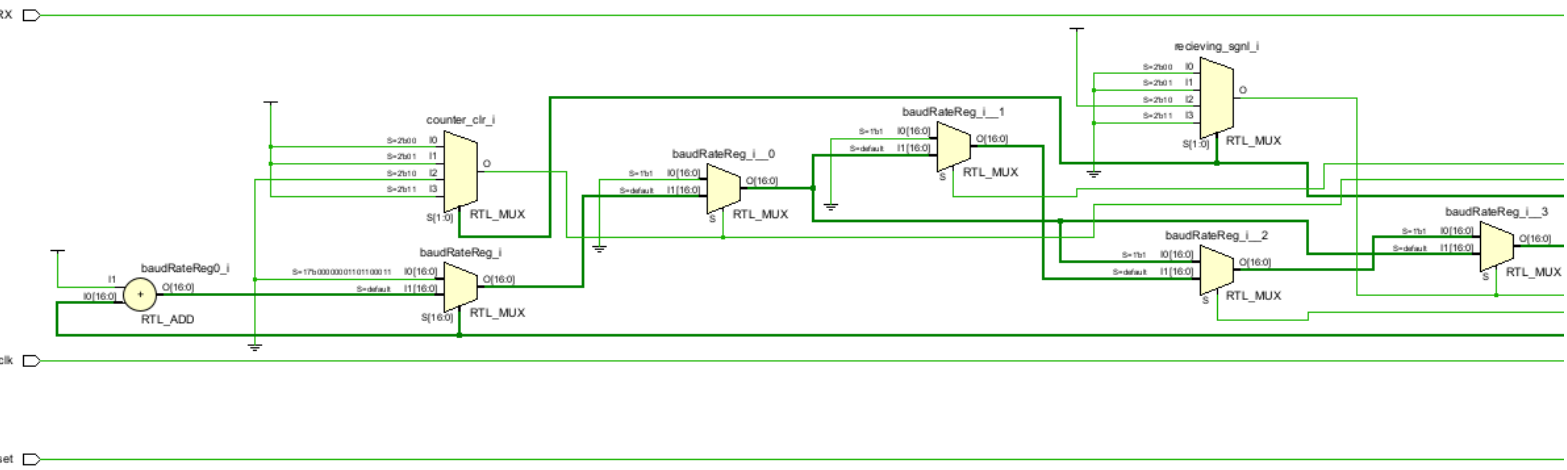

dataTransmit
dataLoad
reset
clk
nStage_i
nStage_i__0
nStage_i__2
nStage_i__1
counter_clr_i
countbit_rnt_9_i
countbit_rnt_9_reg
pStage_reg[1:0]
baudRateReg_i__1
baudRateReg_i__0
baudRateReg0_i
baudRateReg_i
RTL_MUX
RTL_REG_SYNC
RTL_ADD

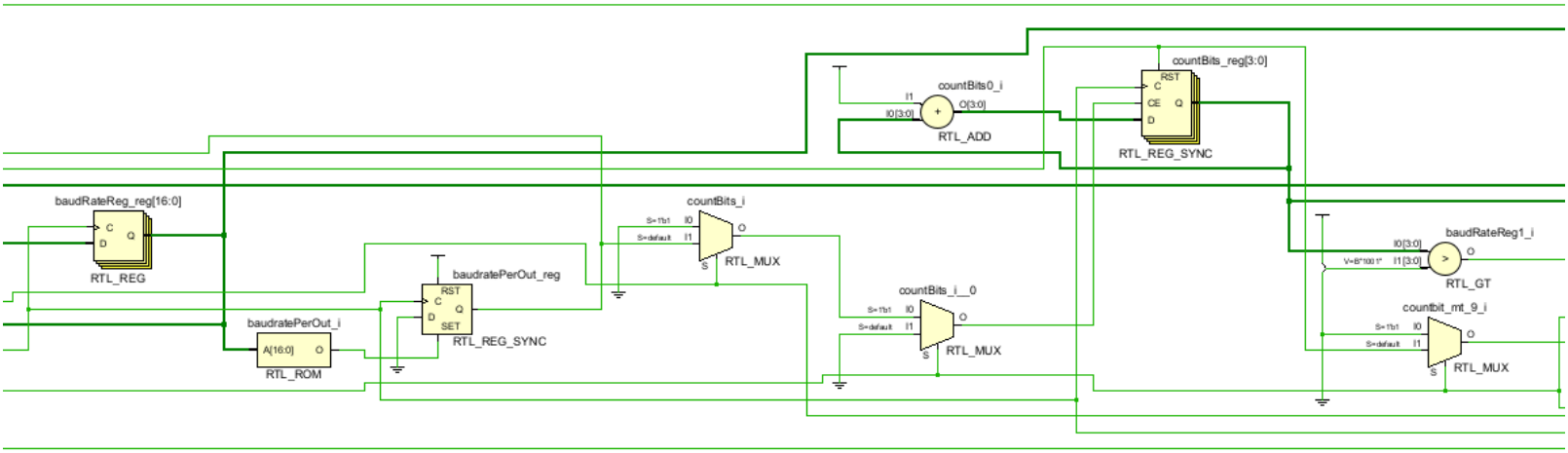
send_sgnl_i
countBits_i
countBits_i__0
baudRateReg_i__2
baudRateReg_i__3
countBits0_i
baudratePerOut_reg
baudratePerOut_i
baudRateReg_reg[16:0]
RTL_MUX
RTL_ADD
RTL_ROM
RTL_REG
RTL_REG_SYNC

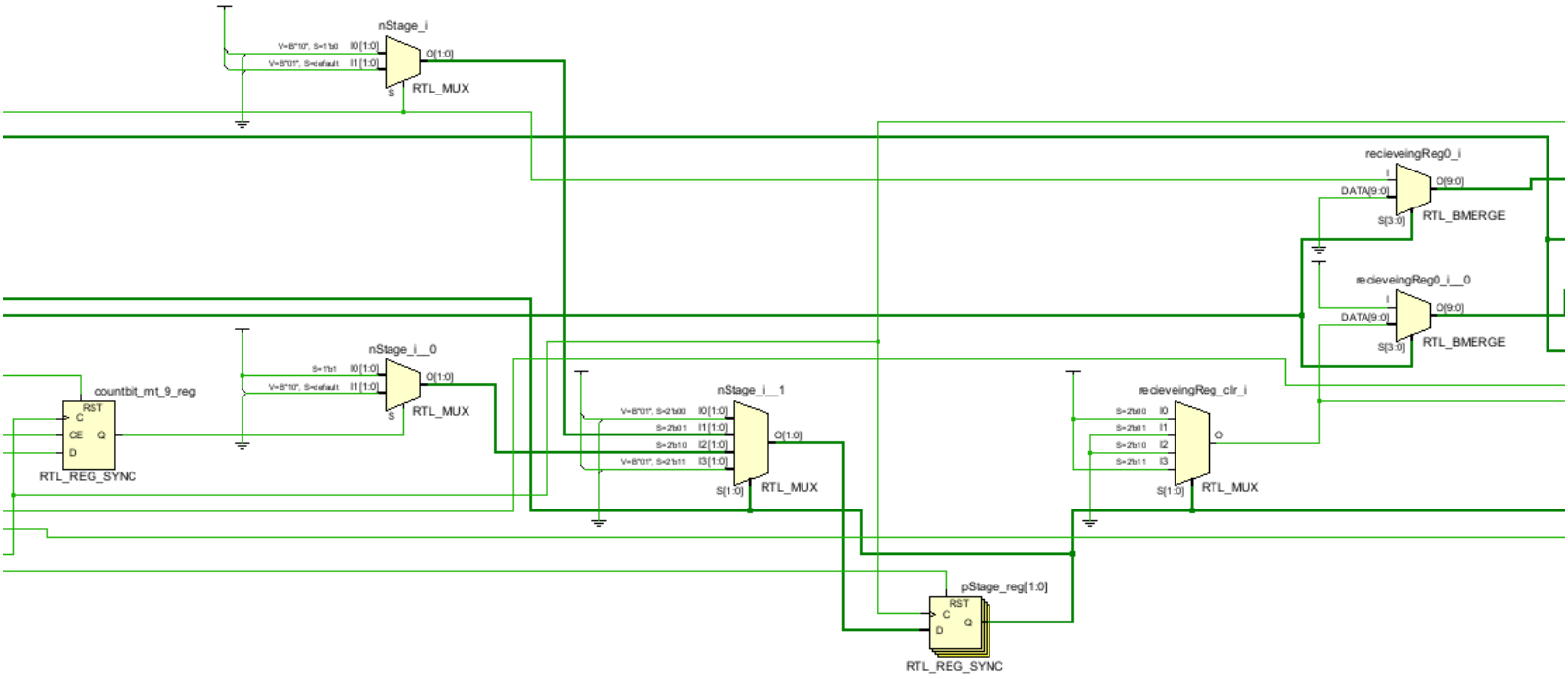

TX_i
RTL_MUX
sendingReg0_i
RTL_RSHIFT
sendingReg_i
RTL_MUX
sendingReg_i__0
RTL_MUX
sendingReg_i__1
RTL_MUX
sendingReg_reg[9:0]
RTL_REG_SYNC
sendingReg_i__4
RTL_MUX
sendingReg_i__5
RTL_MUX
TX_i__0
RTL_MUX
TX_reg
RTL_REG_SYNC
TX
baudRateReg1_i
RTL_GT
countBits_reg[3:0]
RTL_REG_SYNC
sendingReg_i__2
RTL_MUX
sendingReg_i__3
RTL_MUX
txbuf_dr_i
RTL_MUX
txbuf_load_i
RTL_MUX
TXBUF_i__1
RTL_MUX
TXBUF_i__3
RTL_MUX
TXBUF_i__0
RTL_MUX
TXBUF_i__2
RTL_MUX
TXBUF_i
RTL_MUX
TXBUF_reg[1][7:0]
RTL_REG_SYNC
TXBUF[1][7:0]
TXBUF[3][7:0]
TXBUF_reg[3][7:0]
RTL_REG_SYNC
TXBUF_reg[0][7:0]
RTL_REG
outLight[7:0]
TXBUF[0][7:0]
TXBUF[2][7:0]
TXBUF_reg[2][7:0]
RTL_REG_SYNC

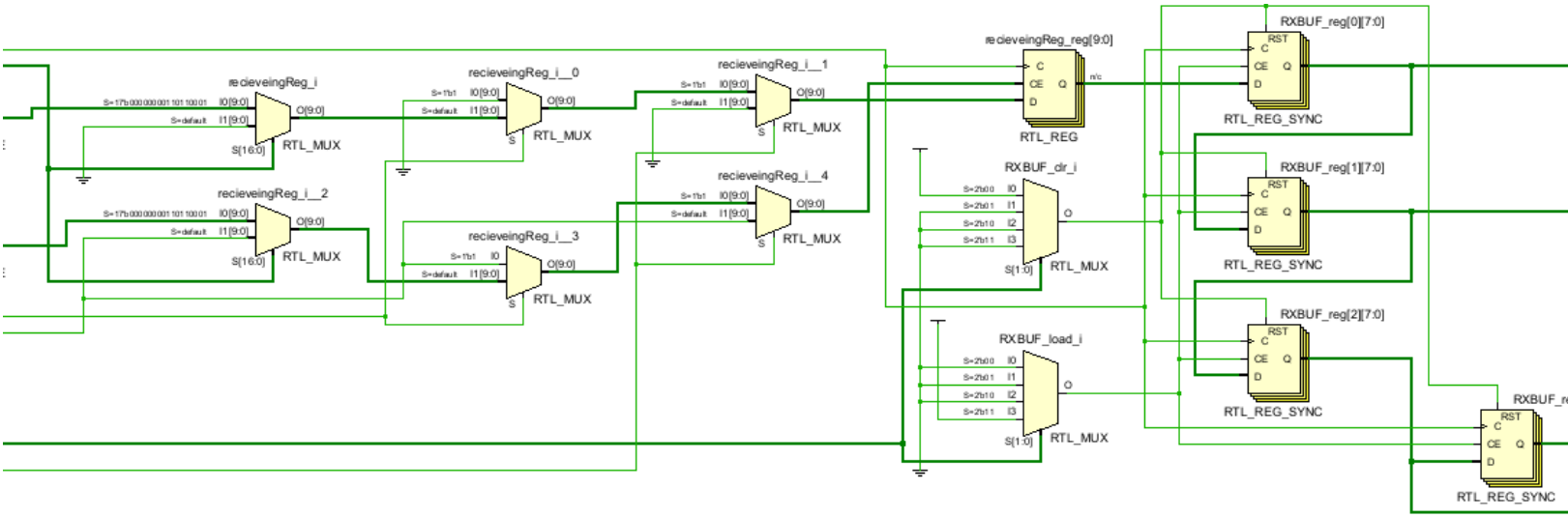

RX State Diagram
Reset
S0
init
rxbuf_clr := 1
recieve_sgnl := 0
counter_clr := 1
countbitCompare9_clr := 1
recievingReg_Clr := 1
baudrateReg_Clr := 1
S1
wait
RX
counter_clr := 1
countbitCompare9_clr := 1
baudRateReg_clr
S2
Recieve
countbitCompare9
RX
recieve_sgnl := 1
S3
Load
countbitCompare9
rxbuf_load := 1
counter_clr := 1
countbitCompare9_clr := 1
recievingReg_clr := 1
baudRateReg_clr := 1

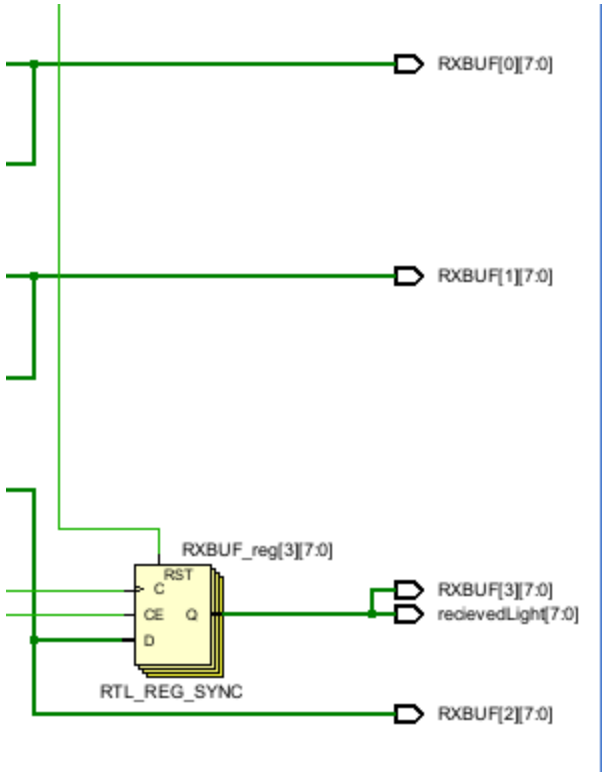

rst
clk
Rx
Reciever
Module
8
3x8
Recv_right
RxBUF

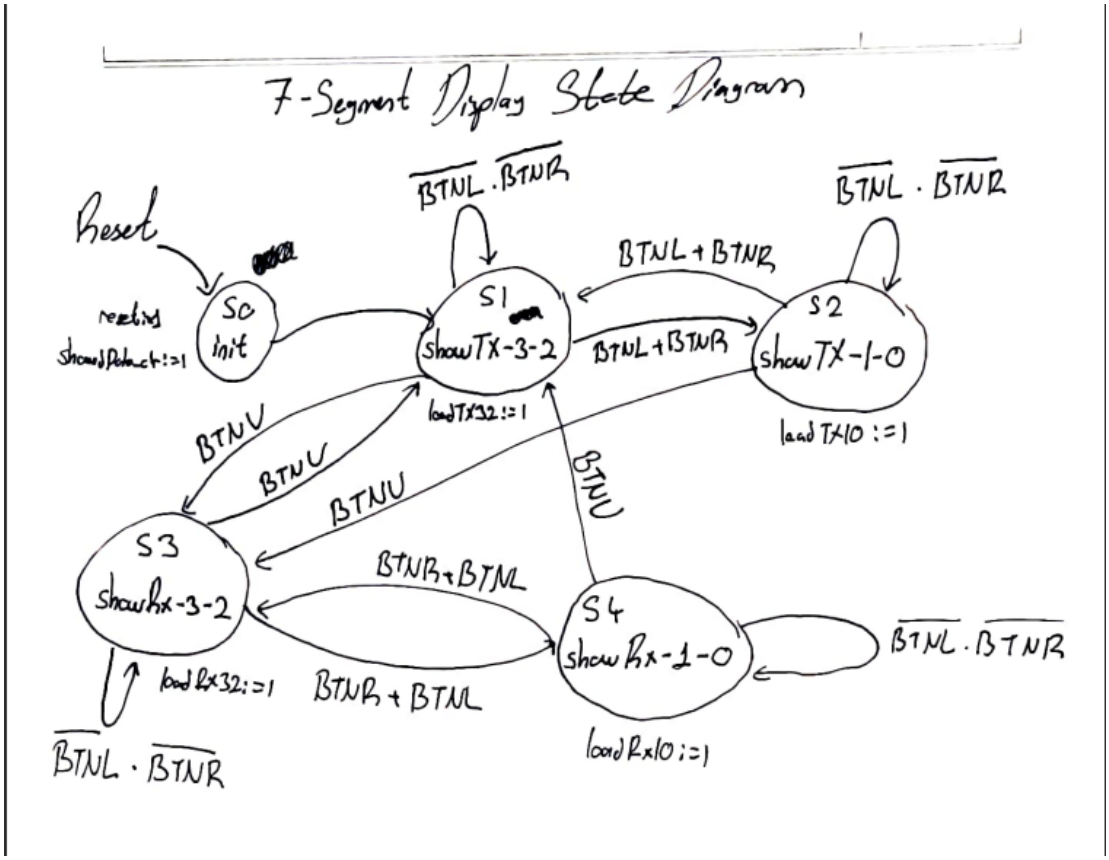

Rx
clk
set
counter_clr_i
baudRateReg0_i
baudRateReg_i
baudRateReg_i__0
baudRateReg_i__1
baudRateReg_i__2
baudRateReg_i__3
recieving_sgnl_i
RTL_ADD
RTL_MUX

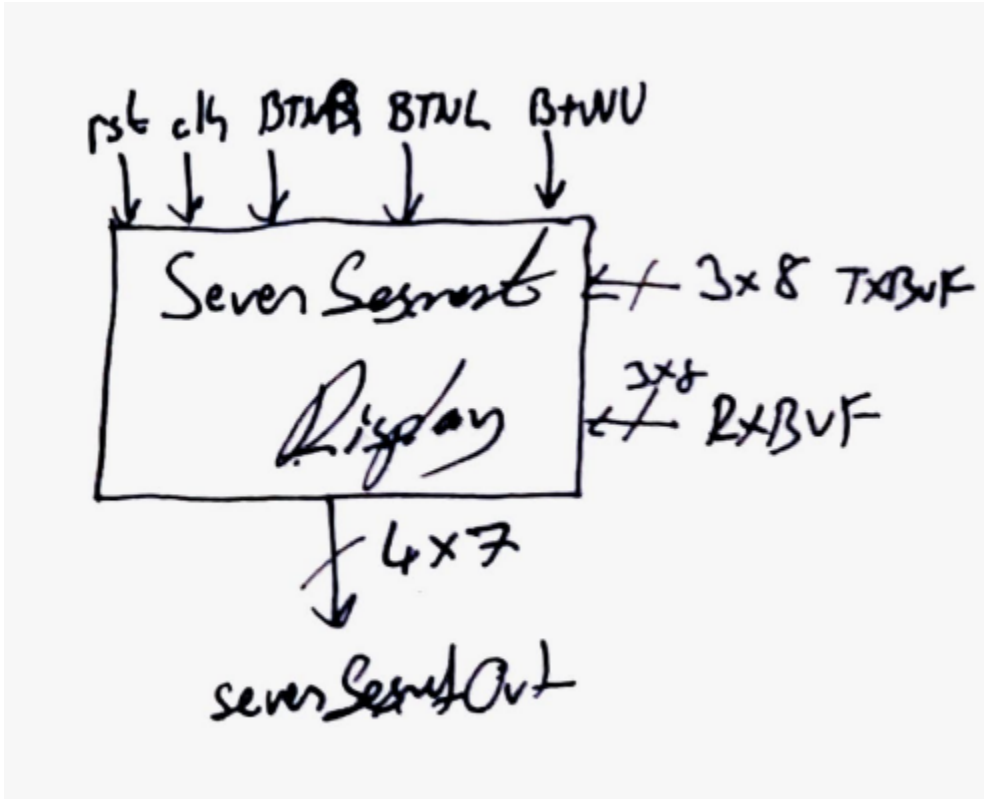

countBits0_i
I1
I0[3:0]
O[3:0]
RTL_ADD
countBits_reg[3:0]
RST
C
CE
Q
D
RTL_REG_SYNC
baudRateReg_reg[16:0]
C
D
Q
RTL_REG
countBits_i
S=1'b1
I0
S=default
I1
O
S
RTL_MUX
baudratePerOut_reg
RST
C
D
Q
SET
RTL_REG_SYNC
baudratePerOut_i
A[16:0]
O
RTL_ROM
countBits_i__0
S=1'b1
I0
S=default
I1
O
S
RTL_MUX
baudRateReg1_i
I0[3:0]
V=B"1001"
I1[3:0]
>
O
RTL_GT
countbit_mt_9_i
S=1'b1
I0
S=default
I1
O
S
RTL_MUX

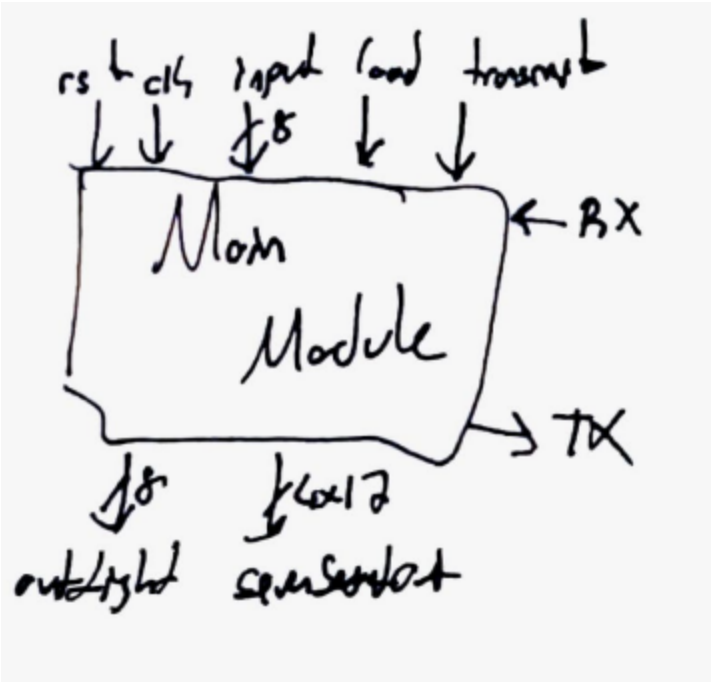

nStage_i
V=B"10", S=1'b0
I0[1:0]
V=B"01", S=default
I1[1:0]
O[1:0]
S
RTL_MUX
recieveingReg0_i
I
DATA[9:0]
O[9:0]
S[3:0]
RTL_BMERGE
recieveingReg0_i__0
I
DATA[9:0]
O[9:0]
S[3:0]
RTL_BMERGE
countbit_mt_9_reg
RST
C
CE
Q
D
RTL_REG_SYNC
nStage_i__0
S=1'b1
I0[1:0]
V=B"10", S=default
I1[1:0]
O[1:0]
S
RTL_MUX
nStage_i__1
V=B"01", S=2'b00
I0[1:0]
S=2'b01
I1[1:0]
S=2'b10
I2[1:0]
V=B"01", S=2'b11
I3[1:0]
O[1:0]
S[1:0]
RTL_MUX
recieveingReg_clr_i
S=2'b00
I0
S=2'b01
I1
S=2'b10
I2
S=2'b11
I3
O
S[1:0]
RTL_MUX
pStage_reg[1:0]
RST
C
D
Q
RTL_REG_SYNC


recieveingReg_i
recieveingReg_i__0
recieveingReg_i__1
recieveingReg_i__2
recieveingReg_i__3
recieveingReg_i__4
RTL_MUX
recieveingReg_reg[9:0]
RTL_REG
RXBUF_clr_i
RXBUF_load_i
RXBUF_reg[0][7:0]
RXBUF_reg[1][7:0]
RXBUF_reg[2][7:0]
RTL_REG_SYNC


RXBUF[0][7:0]
RXBUF[1][7:0]
RXBUF_reg[3][7:0]
RXBUF[3][7:0]
recievedLight[7:0]
RXBUF[2][7:0]
RTL_REG_SYNC


7-Segment Display State Diagram
Reset
reading
showSelect:=1
S0
init
S1
showTX-3-2
loadTX32:=1
S2
showTX-1-0
loadTX10:=1
S3
showRx-3-2
loadRx32:=1
S4
showRx-1-0
loadRx10:=1
BTNL . BTNR
BTNL + BTNR
BTNU
BTNR + BTNL


rst
clk
BTNR
BTNL
BTNU
Seven Segment
Display
3x8 TXBUF
3x8 RXBUF
4x7
sevenSegOut


Main
Module
RX
TX