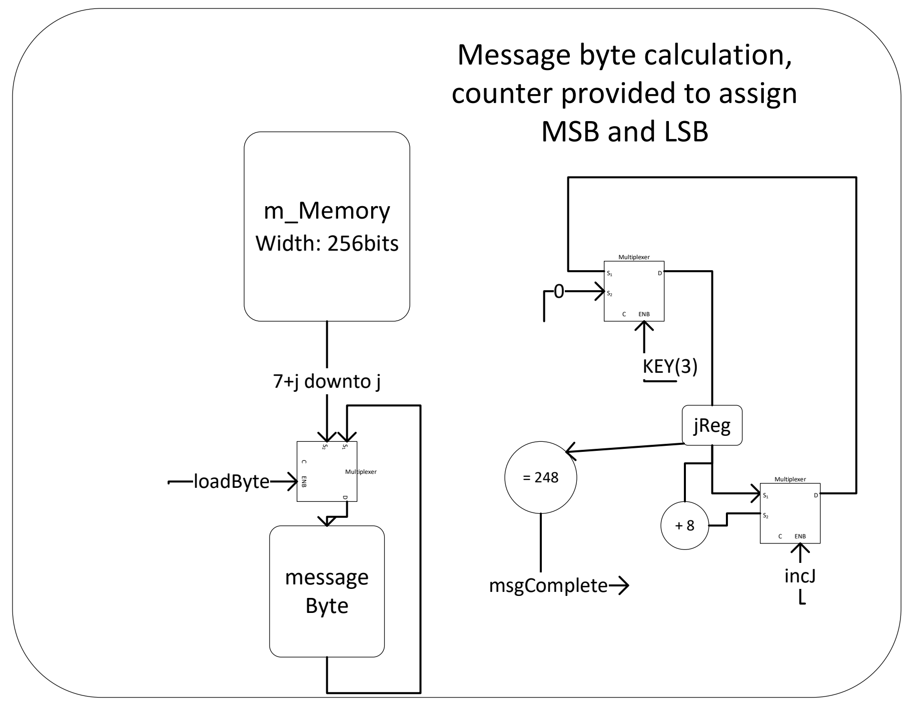
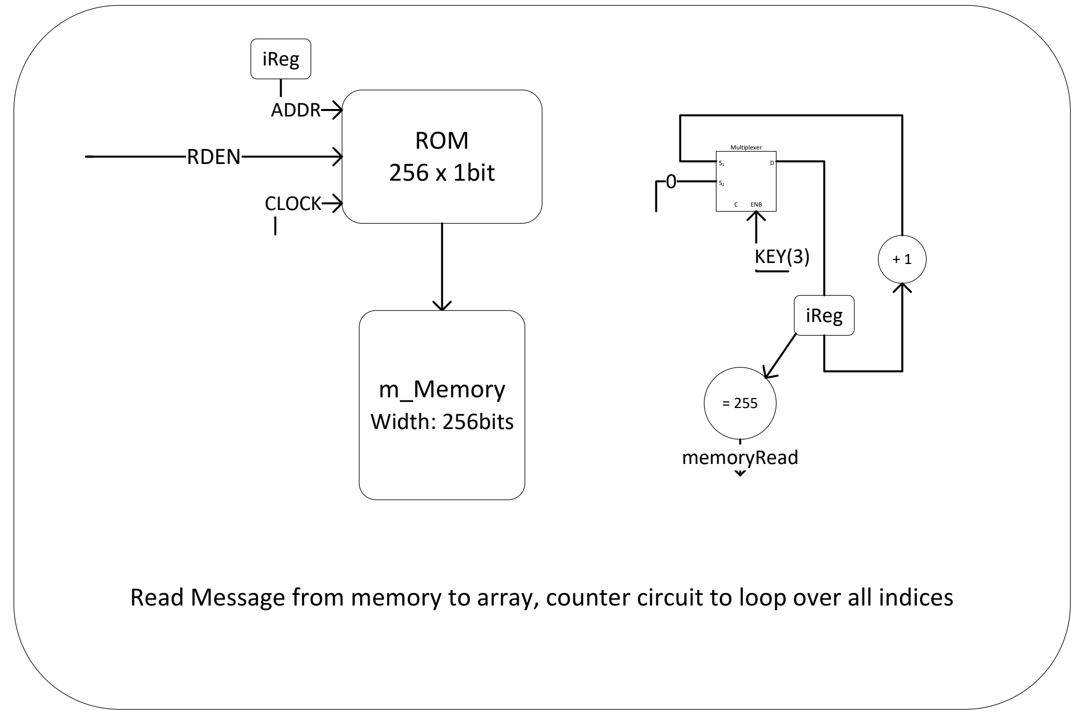
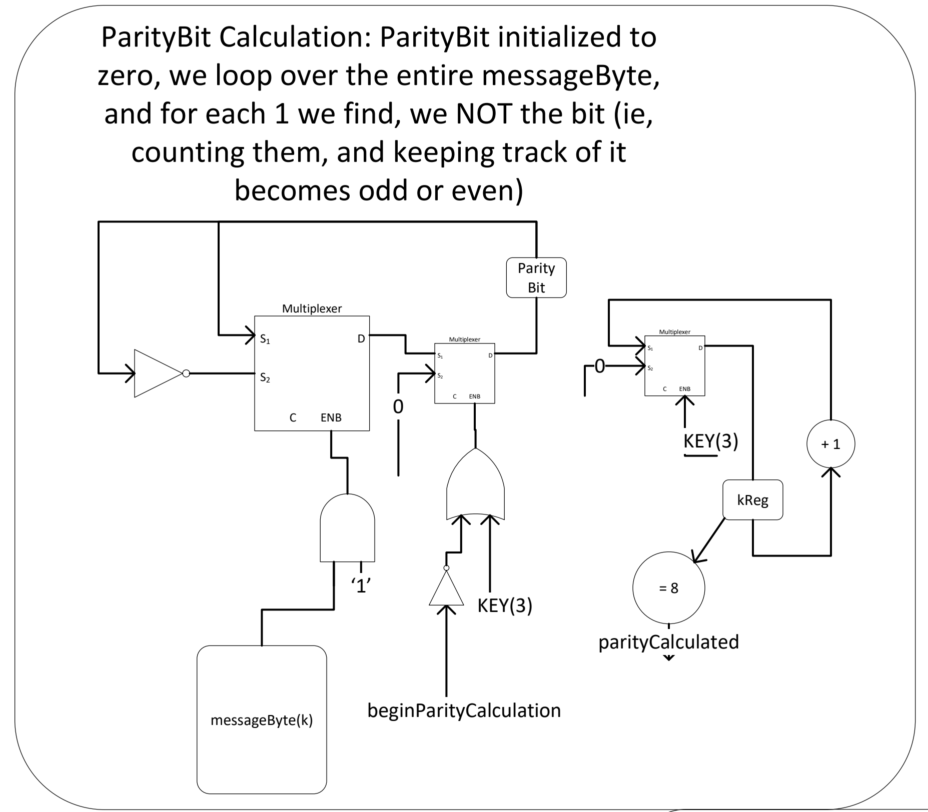
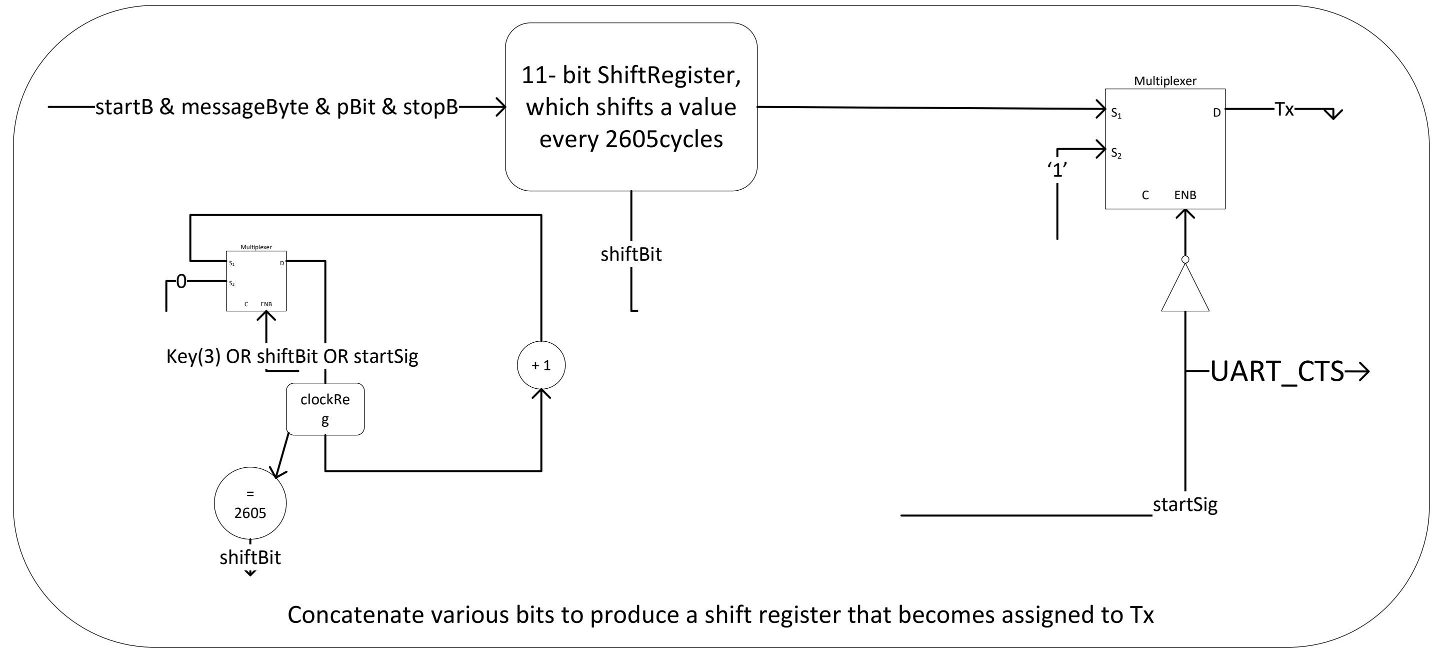
Transmitter RTL

Datapath


Message byte calculation, counter provided to assign MSB and LSB


Read Message from memory to array, counter circuit to loop over all indices


ParityBit Calculation: ParityBit initialized to zero, we loop over the entire messageByte, and for each 1 we find, we NOT the bit (ie, counting them, and keeping track of it becomes odd or even)


Concatenate various bits to produce a shift register that becomes assigned to Tx
startB & messageByte & pBit & stopB
11- bit ShiftRegister, which shifts a value every 2605cycles
shiftBit
Multiplexer
Tx
'1'
UART_CTS
startSig
0
Key(3) OR shiftBit OR startSig
clockReg
+ 1
= 2605
shiftBit

Startbit, and Stopbit vectors
startB <= '0';
stopB <= '1';

50Mhz/19200baud
= 2605cycles/bit

Note: There is some abstraction in logic in the provided Datapath, since too much detail would not be beneficial to provide. Though, the overall operations, and signals from the FSM are included. Explanations are provided in the States chart for ease of understanding.

Group 28: Rajnesh Joshi & Devon Sandhu

FSM

States

INIT
Description: State will begin all our counters to zero, and begin the process of transmitting the message held in memory

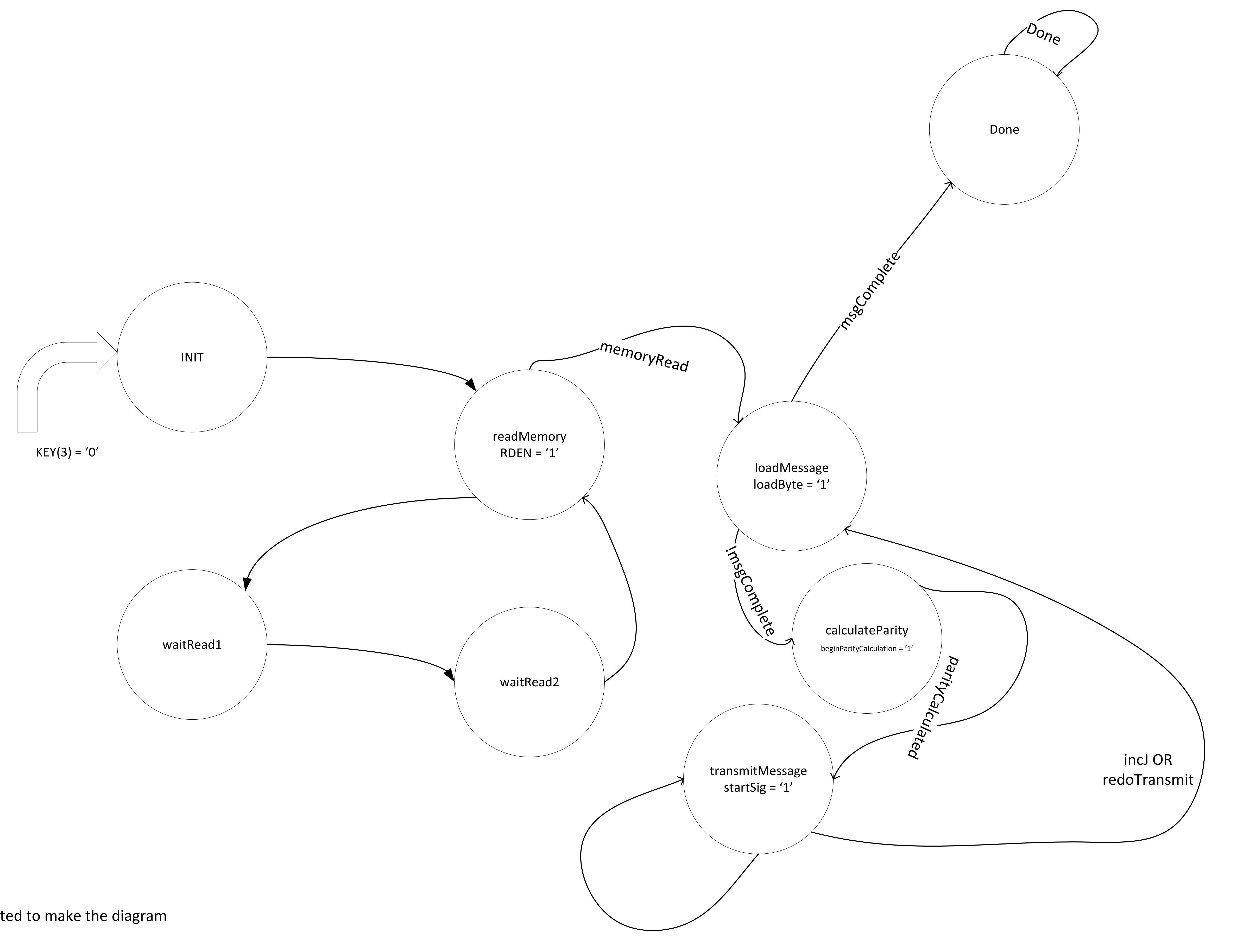
readMemory
Description: This state is when we load an array with the contents held in ROM. There are two additional necessary states that follow this, **waitRead1** and **waitRead2** for timing purposes.
FSM Input: memoryRead
Datapath Input: RDEN

loadMessageByte
Description: Loads a byte from the messge array into a 8-bit signal, after each successful transmission, we load the next byte.
FSM Input: msgComplete
Datapath Input: loadByte

calculateParity
Description: Loop over the entire message byte and calculate the parity. NOT the parity after each high value found in the message byte. It will also increment J for the next message transmission.
FSM Input: parityCalculated
Datapath Input: beginParityCalculation

transmitMessage
Description We raise UART_CTS high for handshaking, and append the startBit, messageByte, parityBit, and stopBit into an 11-bit shift register. If UART_RTS is also high on the reciever's end, then we take the output of this register as an internal signal and map it to the output UART_TX. We hold each bit for a total of 2605 clock cycles. Once the message is completely transmitted, we return to load the next message, or the **Done** state based on the obvious condition of if we are done or not.
FSM Input: transmitDone
Datapath Input: startSig, incJ

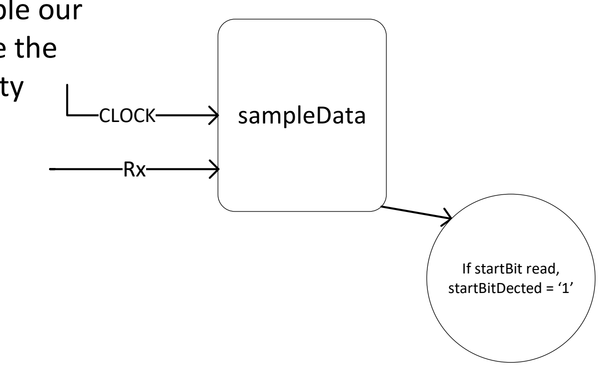
Note: All clocks take CLOCK_50, that is omitted to make the diagram look more clean.

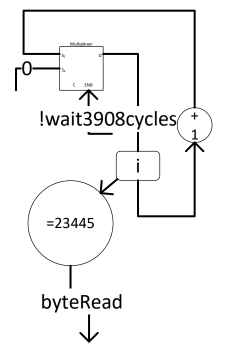


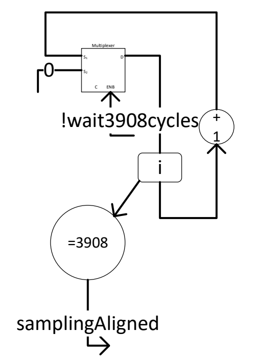
Receiver RTL

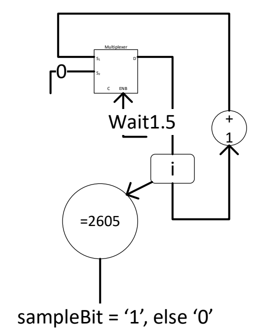
Quite frankly, I had a bit of trouble coming up with the Datapath on the Receiver end. Particularly, I am slightly unsure on how to accurately depict a circuit for sampling data. I have included counters that send signals at appropriate times for when to sample our data on the data frame, and made the detection mechanism, and parity verification clarified.

Once again, explanations in the States table is provided so if the diagrams are hard to follow, please refer to that.

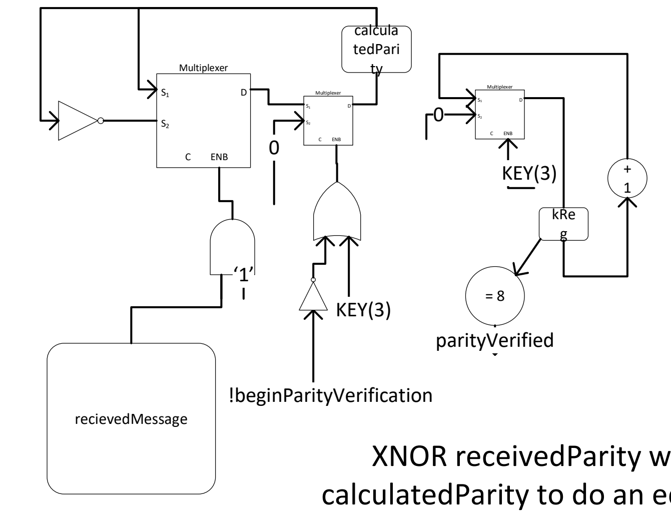








ParityBit Verification



XNOR receivedParity with calculatedParity to do an equality test, and output redoTransmit high or low accordingly.

Group 28: Rajnesh Joshi & Devon Sandhu

50Mhz/19200baud
= 2605cycles/bit

States

IDLE

Description: Sample data at the internal circuits clock frequency. Once the Rx line is pulled low, in other words, a start-bit is detected, we reduce our sampling frequency to match the baud rate. In this state, UART_RTS is on which tells our transmitter that we are ready to receive data.

Wait1.5Periods

Description: This state simply waits 1.5 bit/periods, or 3908 clock cycles, so that when we begin sampling our data, we do so at the middle of each bit transmitted which gives us better data integrity.

readData

Description: Begin reading data at 1bit/periods, or 2605 clock cycles, for a total of 9 bits.

verifyParity

Description: Turn off UART_RTS while we verify that the parity bit that was read is valid, if not, signal redo to the Transmitter FSM to resend the same message and discard this one.

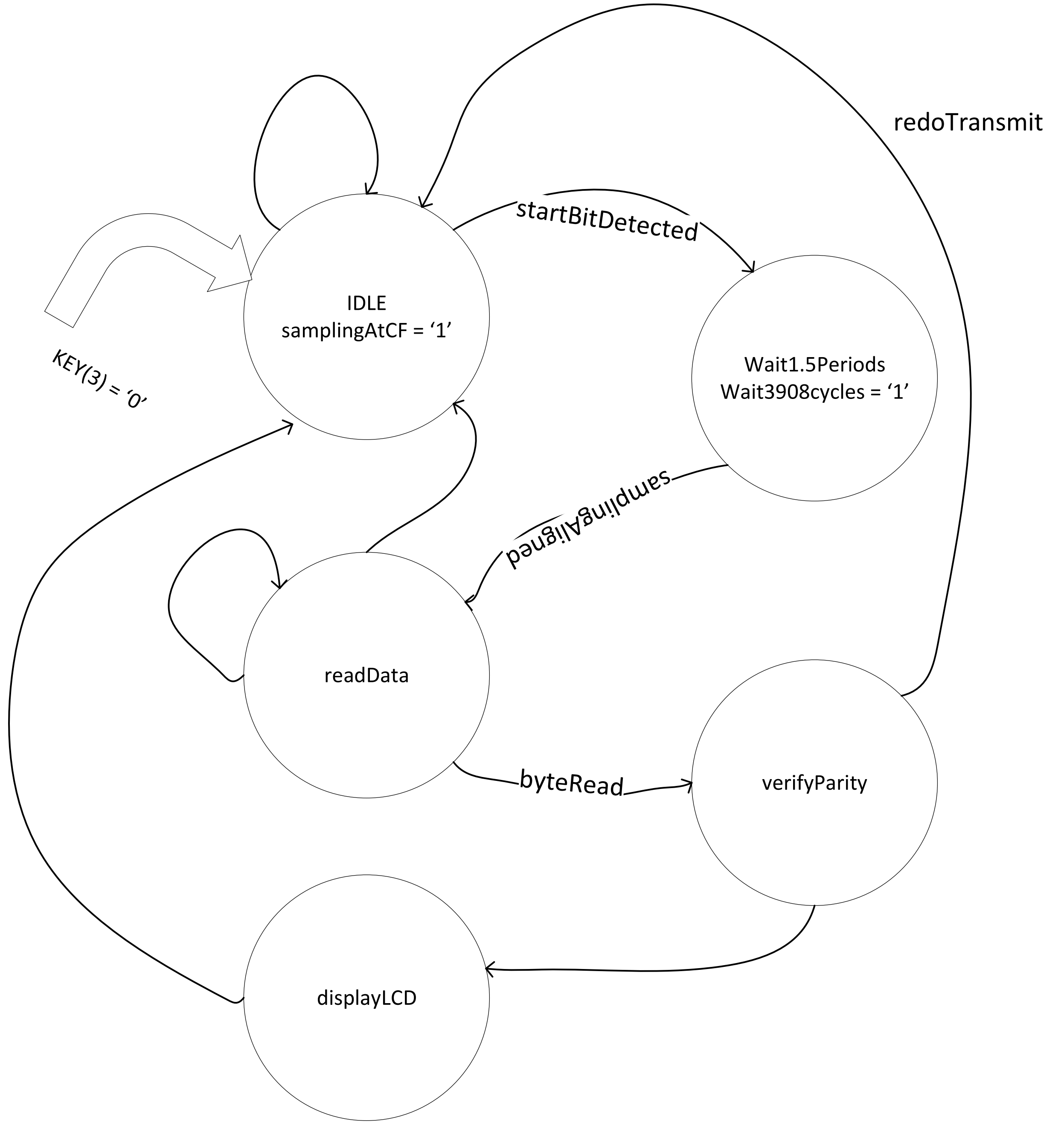
displayLCD

Description: We display the byte on the LCD

transmitMessage

Description: Return to the IDLE state, turn on UART RTS and wait for the next byte of data to come in.

Reciever FSM



redoTransmit
startBitDetected
KEY(3) = '0'
IDLE
samplingAtCF = '1'
Wait1.5Periods
Wait3908cycles = '1'
samplingAligned
readData
byteRead
verifyParity
displayLCD