Lab 7: Asynchronous Serial Receiver

Shrey Patel 2019CS10400
Aayush Goyal 2019CS10452

May 2022

Contents

1 Implementation

1.1 Display

- We have used a seven-segment display to display the digits received from the transmitter(which is the gtkterm console in our case). Since only 8 data bits are involved in the data transfer, only two digits are to be displayed on the board.
- Like previous assignments, we have used a counter which updates with the clock cycle. But, since there are only two digits, we check only the 20th bit of the counter for switching between the digits to display. So, now the time period of switching between the digits is 2^{19} clock cycles or 5.2ms.
- The current number to be displayed is stored in the variable ***number*** while the data bits received till now are stored in the variable ***rcv_number***. The received number is transferred to display only after receiving all the bits (start + data + stop), else the incomplete bits are discarded and the number on display doesn't change. The number displayed initially is "00".

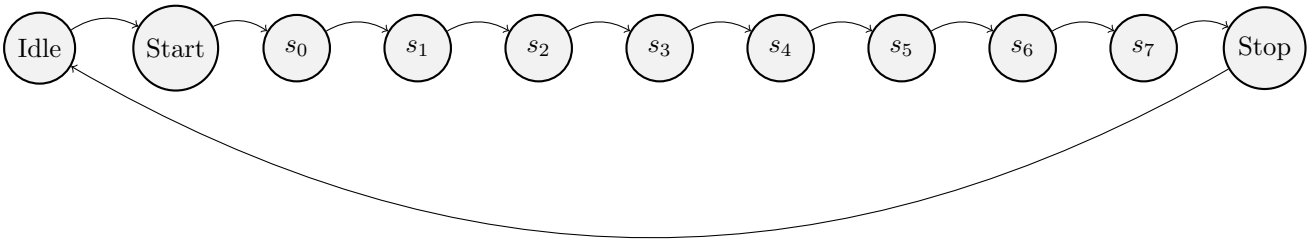
1.2 States

- The current state of the FSM is stored in the variable ***state***. The value of the state and its corresponding interpretation are as listed below:
  1. 9 -> Idle State
  2. 10 -> Start State
  3. 0-7 -> Data States(the state number signifies the position of the expected bit)
  4. 8 -> Stop State

1.3 State Transitions

- Any state transition occurs at the edge of the slow cycle which updates every 651 cycles of the original clock. This is because we want the frequency of the slow clock to be 16 times the baud rate which is 9600. So, the required time period is 10^9 / (16 * 9600) = 6510.41ns, which is roughly 651 cycles of the 10ns clock.

- Additionally, we don't check input or update state at every edge of the slow clock, but after every few slow clock cycles. So, we maintain a counter called ***state_counter*** and now, any state transition will take into consideration both the value of this counter and the input bit.
- The receiver is initially in the idle state and stays in it while it keeps receiving '1'. Other ways in which the receiver can transition to idle state is either when the reset button is pressed or when incomplete number of start or stop bits are obtained while receiving any data from the transmitter.
- In the start state, the receiver expects exactly 8 '0' bits which is checked using the state_counter. If it receives less than 8 '0' consecutive bits, it moves back to the idle state.
- In any data state i.e. (from 0 to 7), the receiver first waits for 15 cycles and then accepts the bit received in the 16th cycle as the next data bit. This data bit is then placed in the corresponding position in the stored number depending on the value of the current state i.e. if current state value is i, then *rcv_number[i]* = data bit.
- After receiving all the data bits, the receiver moves onto the stop state where it again waits for the first 15 cycles before receiving a bit in the 16th cycle. If this bit is 1, then the received number is passed onto the display. Otherwise, the number is discarded. Regardless, the receiver then moves onto the idle state.



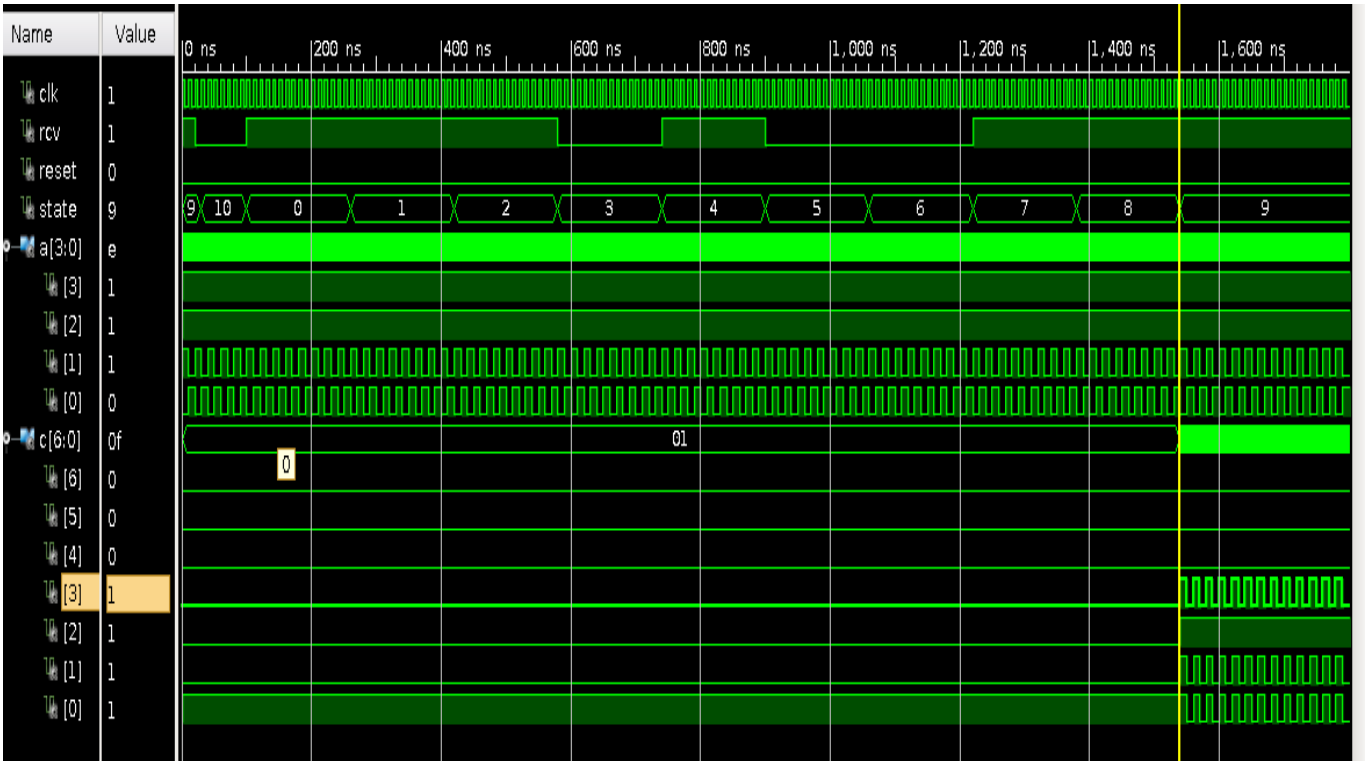
1.4 User Input

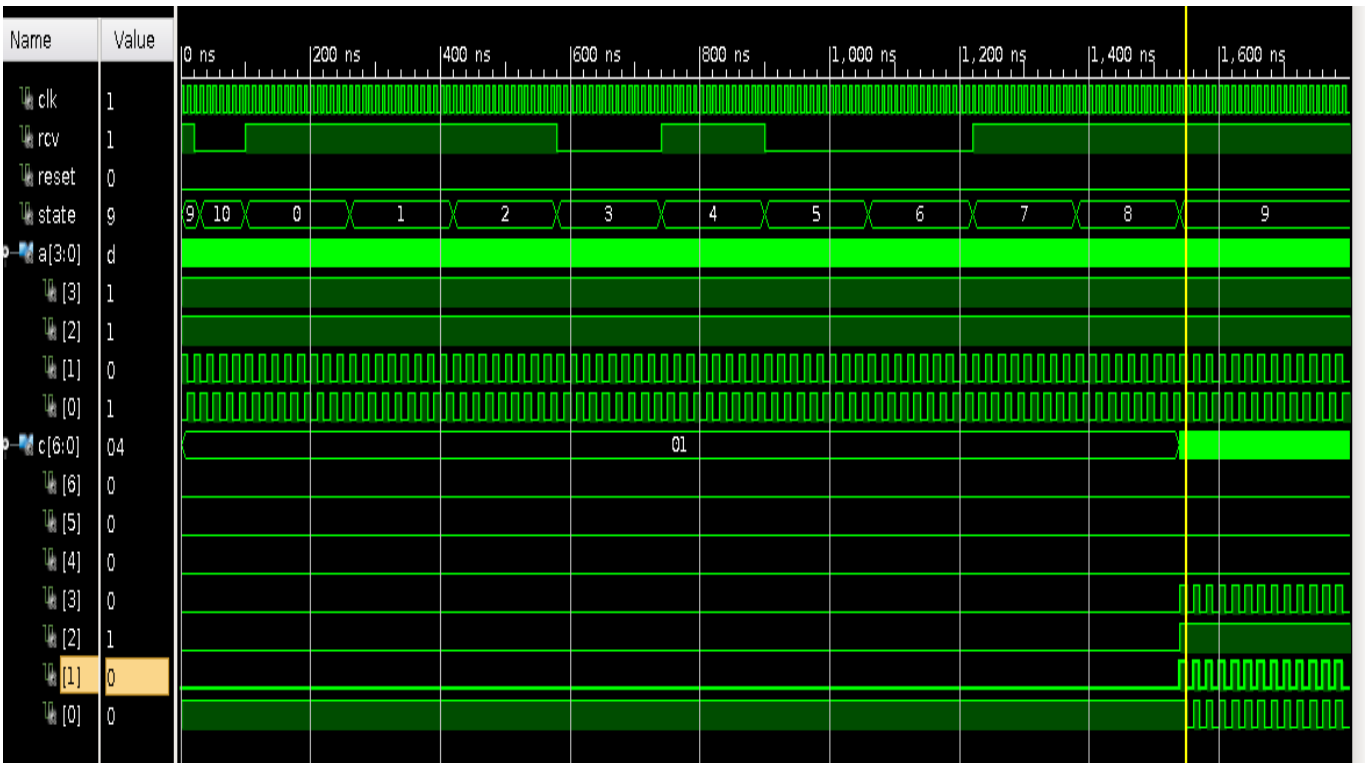
This part involves only one input, the reset button which is mapped to the centre button of the board(i.e. BTNC, V18). On pressing it, the receiver moves back to its idle state, all counters related to the FSM are reset to zero and the digit displayed is also reset to "00".

2 Simulation

2.1 State Transitions

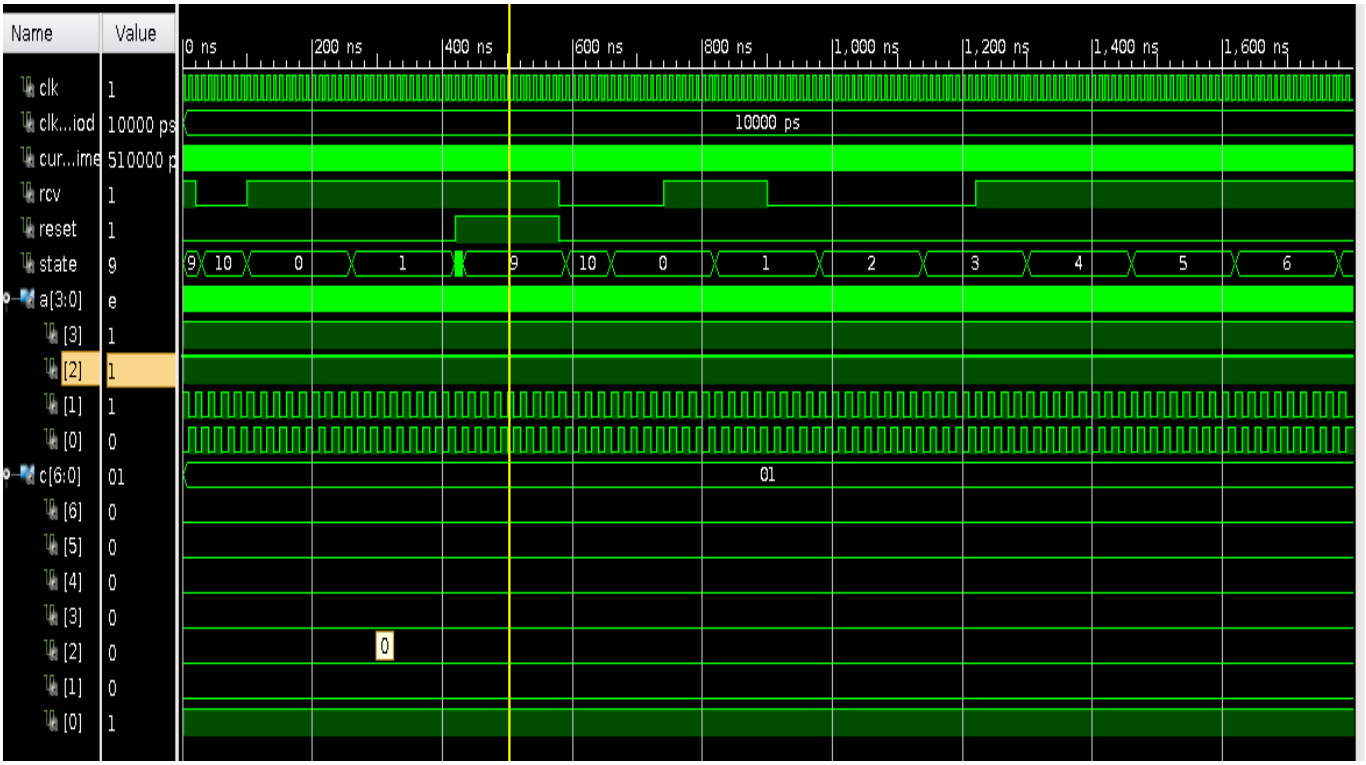
- The first simulation is to check if the receiver obeys the FSM described above.
- Note that the receiver starts in the state 9, moves onto state 10 on reading a '0', then after detecting 8 '0' bits from 20ns to 100ns, moves onto the data state 0.
- From 100ns, the receiver transitions sequentially to other data states after every 16 clock cycles(160ns). The data bits are read in the 16th clock cycle of every such interval. So, the bits read are (after the initial '0' for start state), 11101001 which in reverse is 10010111 or 97.
- The validity of this can be proved after the stop state i.e. state 8, when the cathode value changes suddenly changes which means that the number on display has been changed. The new cathode configuration of the last digit is now 0001111 which is for the digit 7(Figure 1), while that of the first digit is now 0000100 which is for digit 9(Figure 9). So, the new digit on display is 97 as argued above.





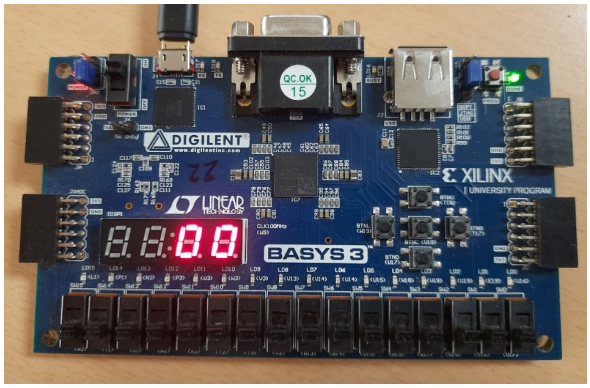
2.2 Reset

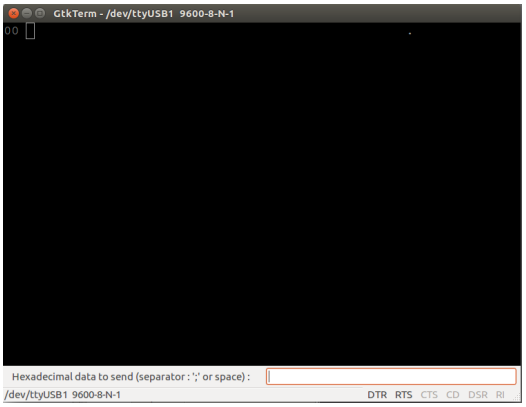
As mentioned before, the reset button moves the FSM to the idle state(9) and discards the partially received bits. It also resets the related counters and resets the display to "00". This can be seen in the simulation below where the reset button is pressed at 420ns, which causes the FSM to move to idle state.

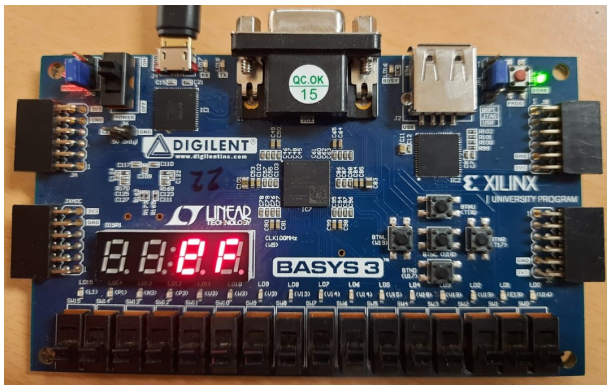


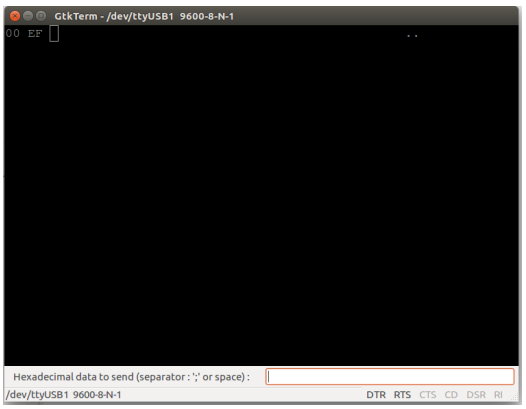
3 Output

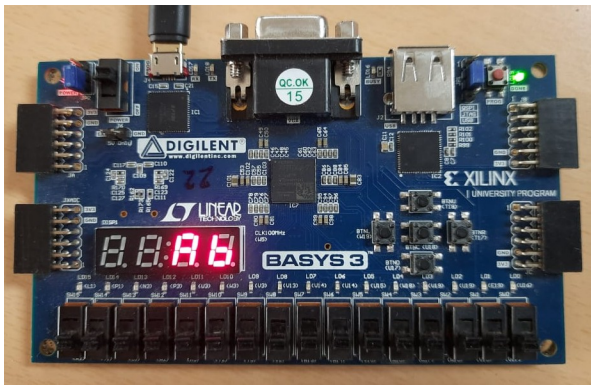
- Order of input: 00 => EF => AB => 56
- Note that the local echo is enabled in each of the below figures, so any input entered can be seen on the gtkterm console.

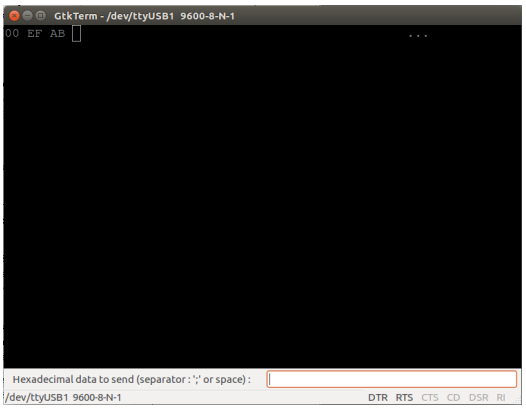


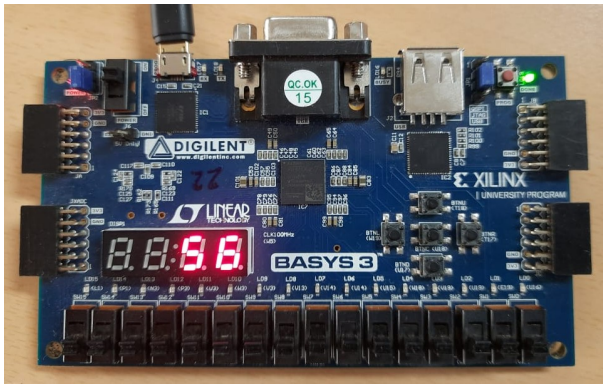


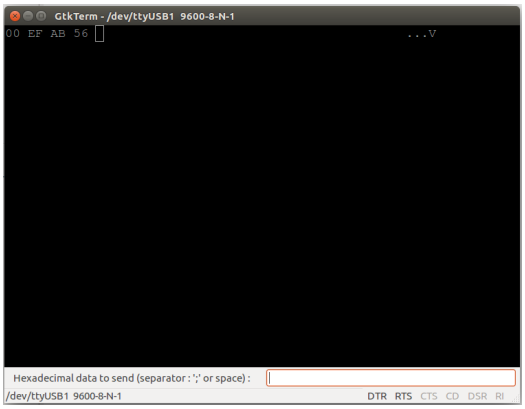












4 Resource Utilization

- **LUTs:** 109
- **Flip Flops:** 111
- **Block RAMs:** 0
- **DSPs:** 0