Lab 10: File Transfer through Serial Receiver/Transmitter

Shrey Patel 2019CS10400
Aayush Goyal 2019CS10452

May 2022

Contents

1 Implementation

1.1 Display

- In this assignment, the display is meant only for debugging purposes. It uses all four digits of the seven segment display as designed in previous assignments.
- The last two digits show the 8-bit number corresponding to the recently received number from the gtkterm to the board. While the first two digits show the 8-bit number recently transmitted from the board to gtkterm. But since all data is now sent as a stream of bytes and all of it is stored/transmitted very quickly as compared to the human persistence of vision, these two numbers are actually seen to be either the latest number read, or the latest number written since the display of intermediate numbers refreshes very rapidly.

1.2 FIFO buffer(Memory)

- As in assignment 9, the BRAM used here has dual ports, one for reading and the other for writing. The size of each cell inside BRAM is now 8 bits and the depth of the buffer is 512 such cells.
- And again, the FIFO buffer used here is fall-through meaning that the first 8-bit number written to it is directly available at the read port without any additional signal. Due to this reason, whenever the first number is written, all the digits of the seven segment display change, while in other cases, the first two digits change only when the 'tx_start' button is pressed.

1.3 Clock Converter

- This part of the FSM generates the clock signals to pass to the other FSM components by using the original 100Mhz clock. It generates two slower clocks: rx_clk of frequency 16 * 9600Hz and tx_clk of frequency 9600 Hz.
- These clocks are manipulated by maintaining separate counters for each of them.
- Additionally, this module also maintains the multiplexing of the display, which is done every 2^18 cycles of the original clock.

1.4 Receiver

- The receiver module is exactly similar to the one designed in assignment 7 i.e. using a 9600 Hz clock(i.e. rx_clk), it changes its state from idle to start state on receiving a '0' bit. It then moves onto the data states on receiving exactly 8 '0' bits. From then on, it captures every 16th bit and stores the resulting 8-bit number into a local variable(and later to BRAM). Finally, it moves to the stop state in which if it finds the 16th but to be '1', it finally moves back to the idle state, expecting another number.
- The modification in this part, is that after every successful receipt of a number, the receiver signals the timing circuit using the signal 'rx_full' that it now ready to push the received number into the memory.

1.5 Transmitter

- The transmitter module works as in assignment 8, in that it stays in the idle state, sending idle bits (ones), and signalling tx_empty to the timing circuit, meaning that it is ready to transmit as soon as it can read from memory.
- Once it has a number, it sends the bits in the same order as what the receiver received i.e. start bits, then data bits and then the stop bits. The only difference here, is that each of these is now sent only once since now the clock is 16 times slower than in assignment 8 (frequency 9600Hz).

1.6 Timing Circuit

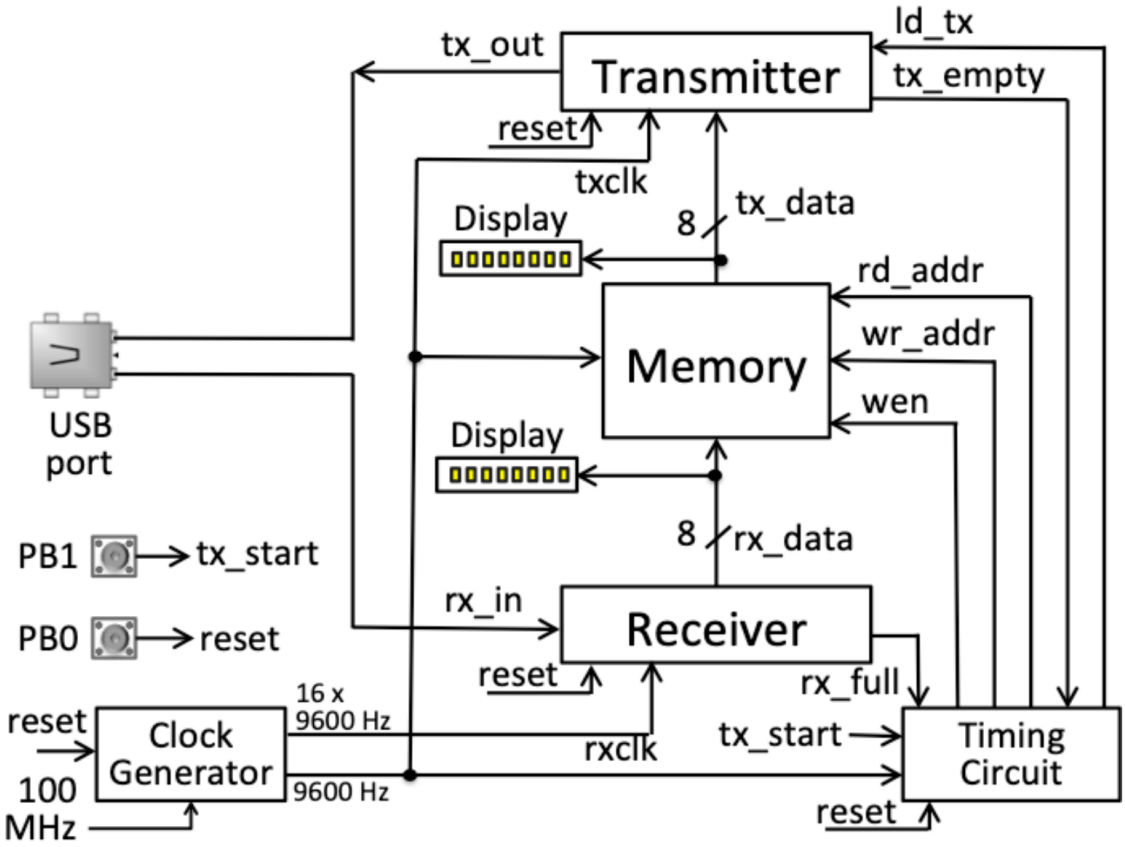
- The timing circuit is the most important component of the FSM as it manages and controls the entire functioning. It manages the input, links the receiver and the transmitter to the memory, stores and passes the head and tail pointers for correctly reading and writing in the memory and finally makes sure that the working of receiver and transmitter are mutually exclusive.
- When the receiver successfully receives a number, it sends the rx_full signal to the timing circuit and if the FSM is currently in the receiver mode, than the timing circuit enables the 'wen' (write enable) signal to push the received number onto the current address pointed to by wr_addr. The receiver disables the rx_full signal once it goes back to the idle state.
- On the other hand, the transmitter continuously sends the tx_empty signal, indicating that it is ready to transmit. Only when the user pushes the tx_start button, the FSM moves on to transmitting mode and the timing circuit enables the ld_tx signal to inform the transmitter module to read the parallel data stored in the memory at the address rd_addr and then send it out serially. When the transmitter starts sending the serial data, is temporarily disables the tx_empty signal to inform the timing circuit that it is busy.

1.7 Handling Input

- There are two inputs in this assignment: tx_start and reset. tx_start is mapped to the center button (U18, BTNC) while the reset button is mapped to the leftmost button(i.e. W19, BTNL).
- The reset button resets all counters in the clock module, brings both the transmitter and the receiver to the idle state, and resets the timing circuit back to the receiver mode while making the head and tail pointer to be same, which signifies empty memory.
- All inputs buttons are debounced. The debouncing has been handled using a slow clock which updates every 2^20 cycles of the original clock.

1.8 FSM

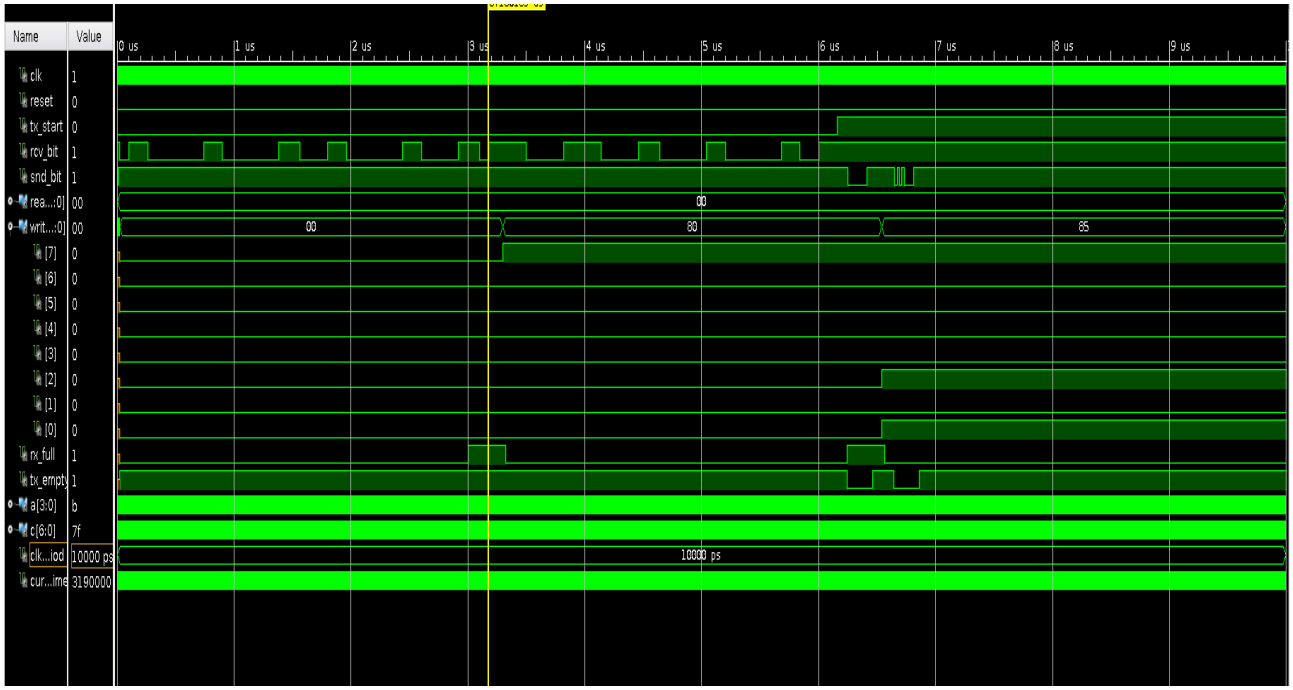
Our FSM exactly matches the sample FSM given, which is as follows:



2 Simulation

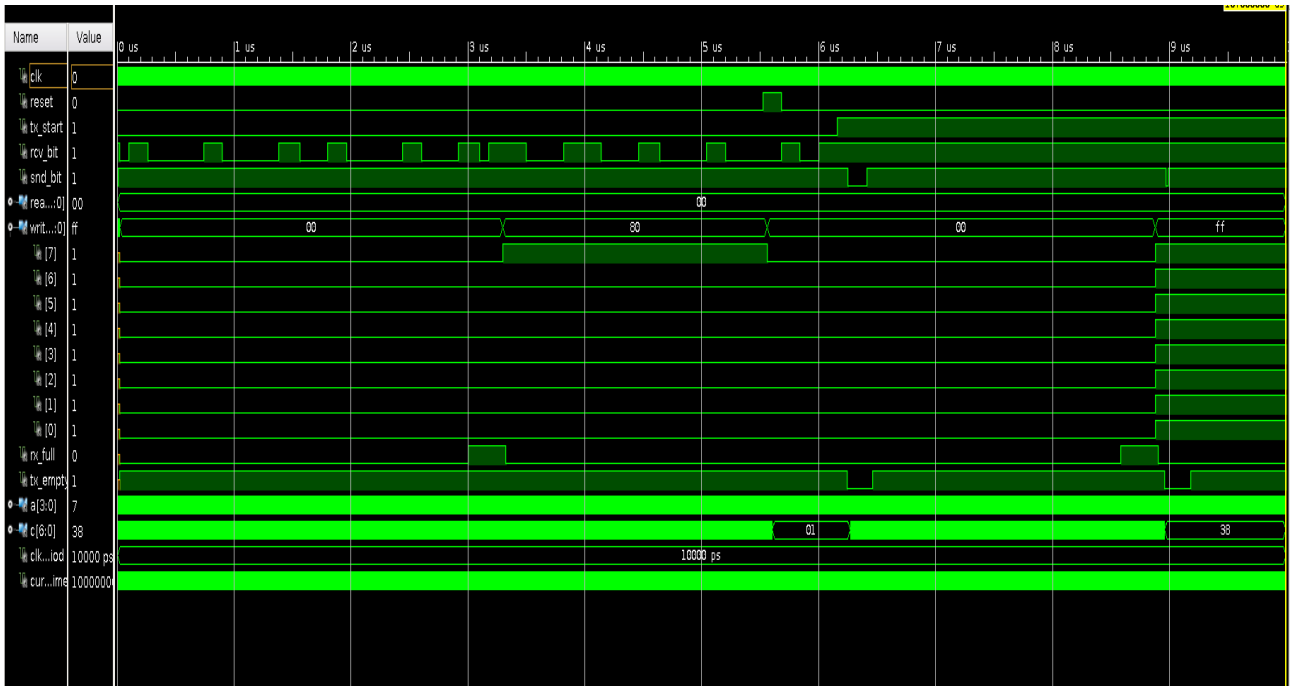
2.1 Mode Switching using tx_start

Pressing the tx_start button informs the timing circuit to start transmitting the data stored in the memory till now. This can be highlighted using the following simulation:



2.2 Reset

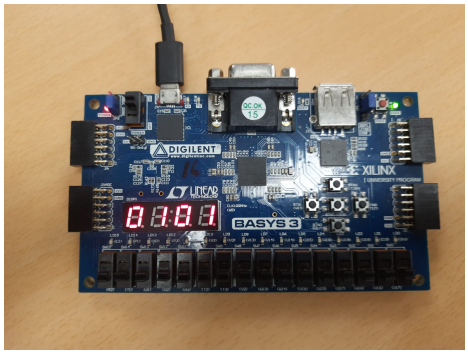
Pressing the reset button resets the counters and re-initialises the values of receiver/transmitter states, read/write numbers, memory pointers, etc. In the simulation below, the reset button is pressed around 5500ns.

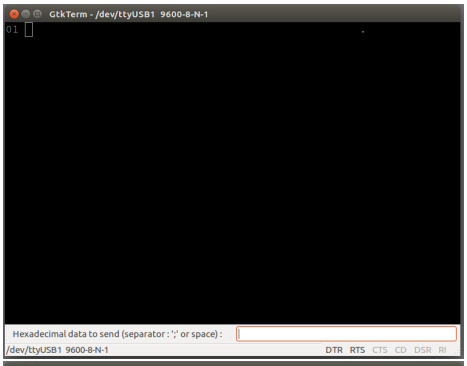


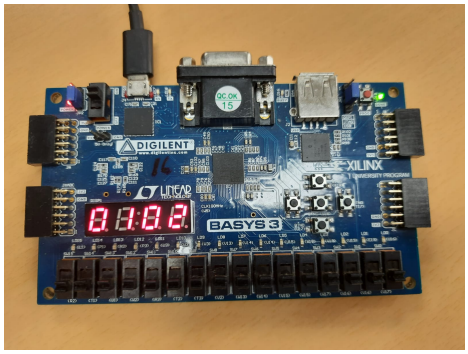
3 Output

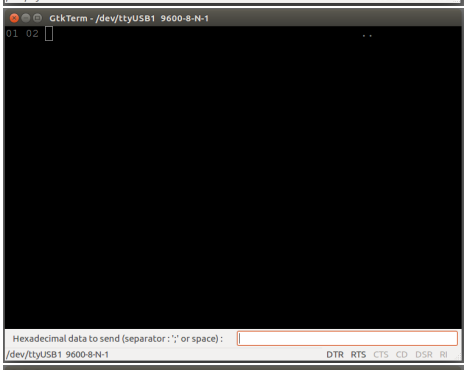
3.1 Receiver Mode

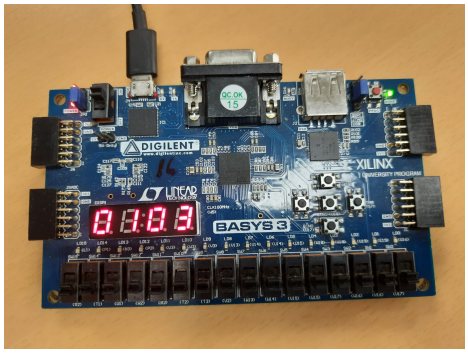
In the images below, we give an input to the board using gtkterm. Since the last two digits of the display correspond to the received number, we can see that it matches the number which is written in the gtk terminal. While the first two terms must remain the same since they change only on reading from the memory. The only exception is the first digit which we discussed is the consequence of the buffer being fall-through.

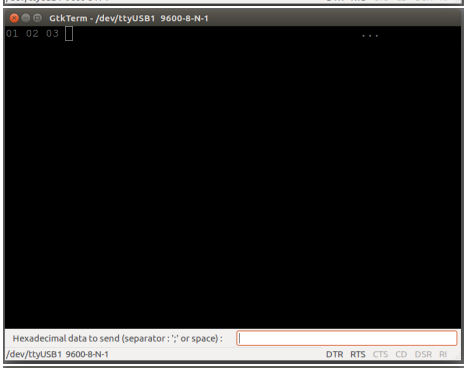


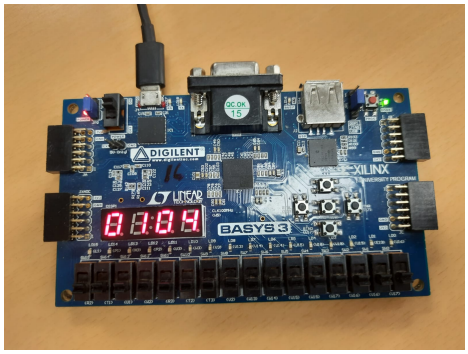


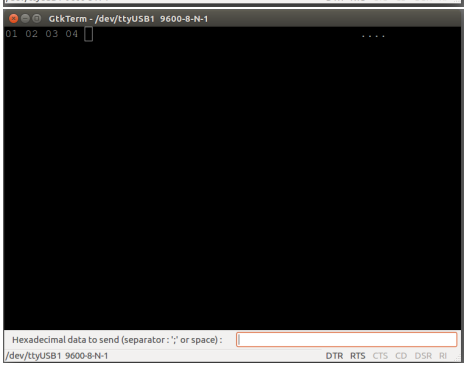






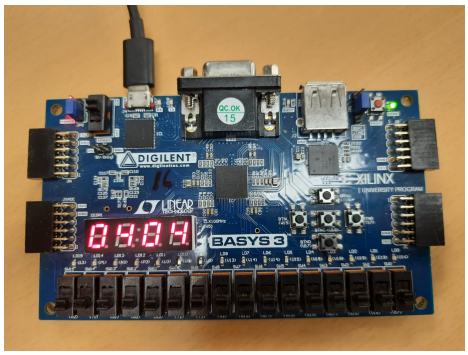


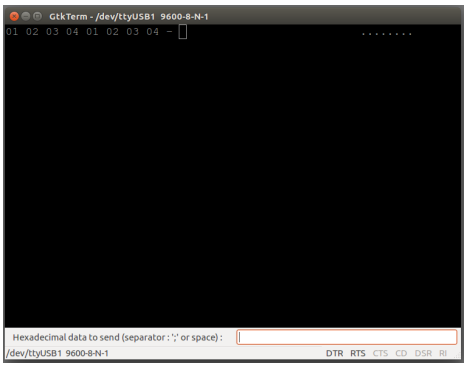




3.2 Transmitter Mode

The images below show the state of the board after pressing the tx_start button, i.e. after switching to transmitter mode. The output in gtk terminal shows that the received sequence of bytes is 01 02 03 04 which is exactly the order in which we sent the bytes. Also, the first two bits on the board display now show 04 which is the most recent number to be read from the memory.





4 Resource Utilization

- **LUTs:** 154
- **Flip Flops:** 241
- **Block RAMs:** 0.5
- **DSPs:** 0