Lab 9: FIFO Buffer

Shrey Patel 2019CS10400
Aayush Goyal 2019CS10452

May 2022

Contents

1 Implementation

1.1 Display

- The display is implemented using the four digits of the seven segment display as designed in previous assignments
- The number corresponds to the last 16-bit number read from the FIFO buffer. Initially, it shows the number "0000".

1.2 FIFO Buffer

- The FIFO Buffer has been implemented using dual port BRAM, of which one port will be used to read the earliest written number while the other port will be used to write a new number into the buffer. Both ports are supplied the same clock.
- The address to read is maintained by the ***tail*** pointer, while the address to write is maintained by the ***head*** pointer.
- Any write operation involves writing into the memory address pointed by the head pointer and then incrementing the head pointer by one, while any read operation involves reading the buffer at the tail pointer and then incrementing the tail pointer by one.
- The buffer is empty if the both the pointers are equal. Now, note that the buffer is circular, meaning that once the head pointer reaches the maximum allowable depth, it circles back to the beginning i.e. zero. We consider the maximum size of the buffer to be one less than the actual maximum depth of the buffer. Here the max depth of buffer is 8 which means the maximum number of digits that can be stored is seven. In this case, the buffer is full if the tail pointer leads the head pointer by one. (Otherwise the emptyness and fullness condition would be the same i.e. head = tail).
- One thing to note is that the FIFO buffer is by default fall-through which means that the first digit written is directly visible on the read port.

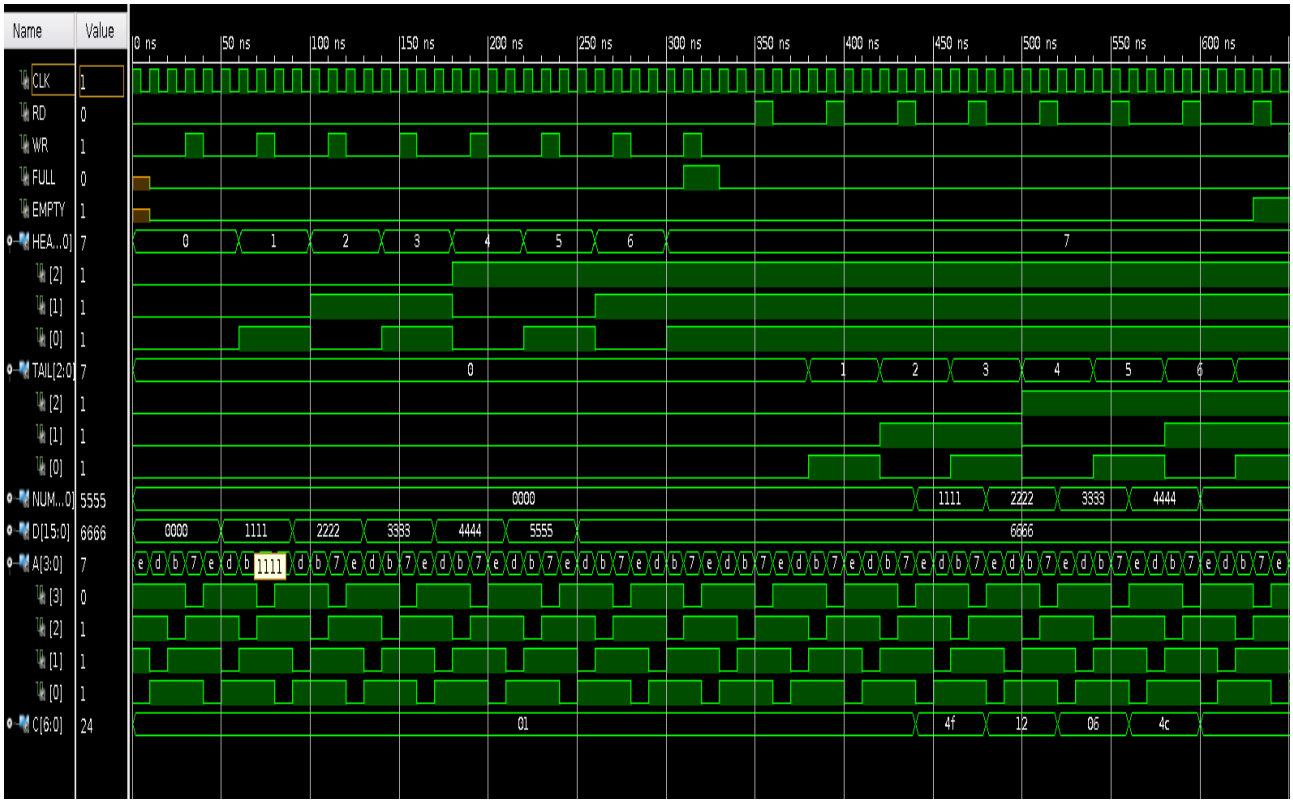
1.3 Handling Input/Output

- There are two types of input handled in this assignment. One is read and the other is write. The read input is mapped to the leftmost button (i.e. W19, BTNL). Pressing it will cause the FSM to fetch the number stored in BRAM at address pointed by 'head' if the buffer is not empty. While the write input is mapped to the centre button (i.e. U18, BTNC) and pressing it will cause the FSM to store the 16-bit input which is indicated by the current configuration of the slide switches, provided that the buffer is not full.
- To indicate whether the buffer is empty or full, there are two LED lights mapped to them. If there is a read input on an empty buffer, then the light LD0(i.e. V16) will turn on. While if there is a write input on a full buffer, then the light LD1(i.e. E19) will turn on. Note that these lights are not toggle-based meaning that they will remain on only for the duration for which the corresponding input button is pressed.
- All inputs buttonss are debounced. The debouncing has been handled using a slow clock which updates every 2^20 cycles of the original clock.

2 Simulation

All the features of this assignment can be explained from the simulation below as follows:

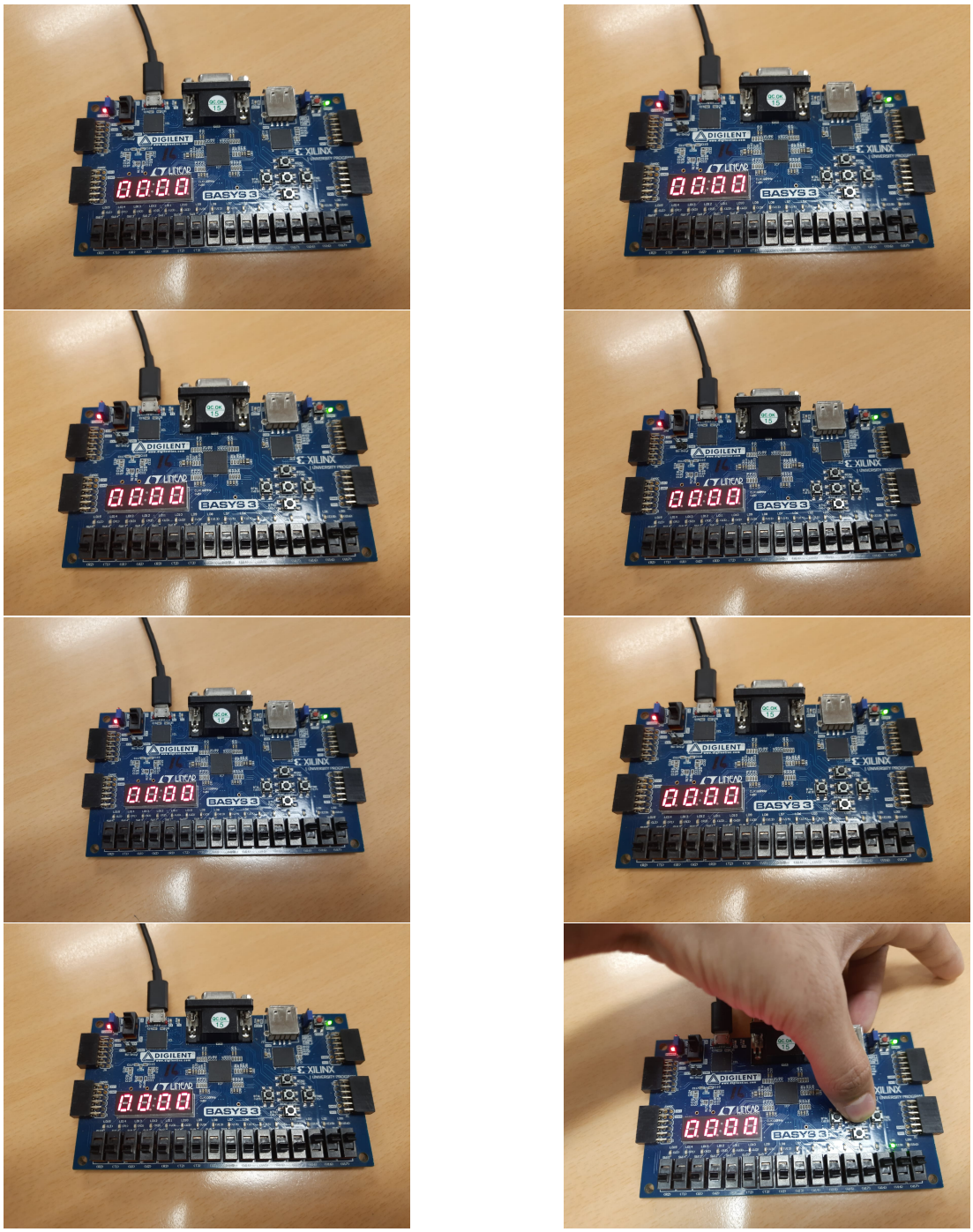
- First of all, we can see the multiplexing of display from the sequence of anode values
- Then, we see the increment of the head pointer whenever the write bit is enabled. The head pointer increases by one every time and once it reaches 7, it cannot increase further since the queue is full. This also triggers the FULL signal as shown in the simulation(at around 310 ns). We also change the value of D periodically to check whether the stored values is correctly read.
- After 350ns, we enable the read bit periodically at every 40ns intervals. Now, we can see the tail bit being incremented by one as expected. But this time, we can also see the value of NUMBER changing, which stores the value read from the memory. Note that the sequence of values assigned to NUMBER: 0000, 1111, 2222 and so on exactly matches the sequence of values provided to D during the writing phase. This confirms that the buffer follows FIFO order. Additionally, we can also see that after he buffer becomes empty, if we press read button, the empty signal gets enables(at about 630ns).



3 Output

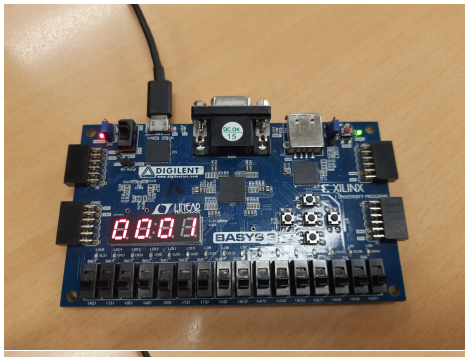
3.1 Input

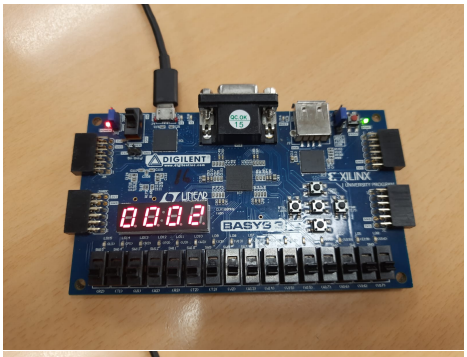
The following images contain the input entered using the write button. Note that after adding 7 numbers, adding the 8th number triggers the full signal

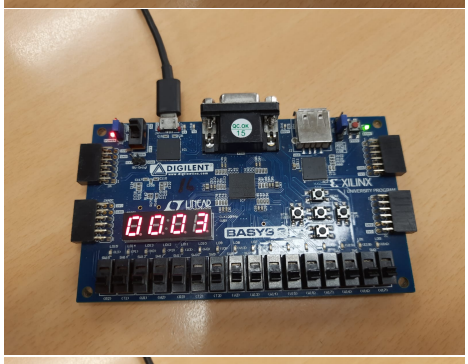


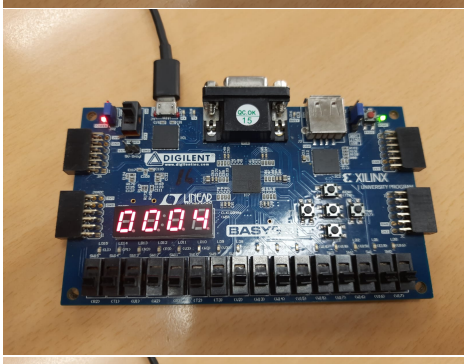
3.2 Output

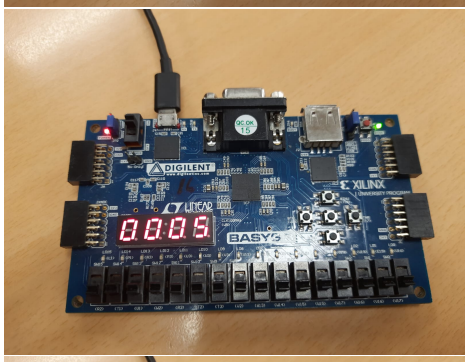
The following images contain the output displayed on the seven segment displays after pressing the read button. Note that the order in which the numbers appear is the same as the order in which the numbers were written. Also, in the last figure, we try to read from an empty buffer which triggers the empty signal.

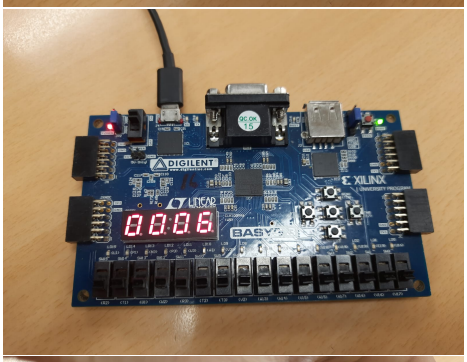


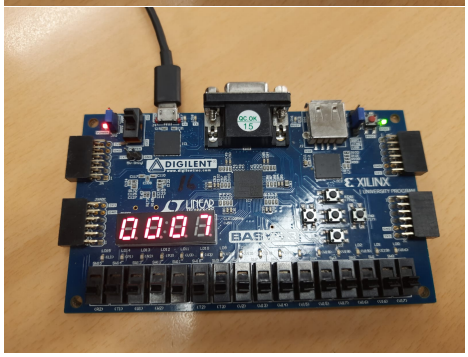


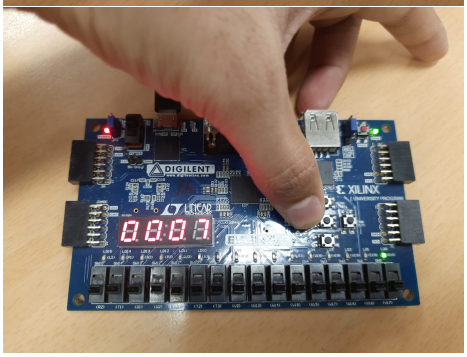












4 Resource Utilization

- **LUTs:** 26
- **Flip Flops:** 74
- **Block RAMs:** 0.5
- **DSPs:** 0