# FIFO BUFFER ON FPGA

AELLA ABHILASH REDDY (EE16BTECH11001)

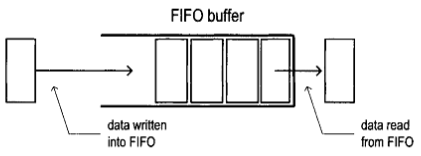
April 27, 2019

## 1 OBJECTIVE

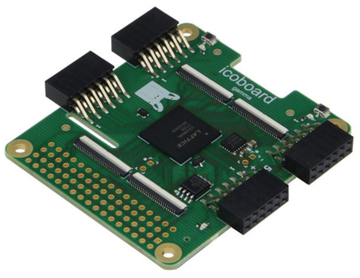
The aim of the project is to implement an FIFO(First In First Out) Buffer on FPGA. The FIFO module takes input data when write is enabled, displays the output data when read is enabled. It also indicates whether FIFO is full or not, empty or not.

## 2 INTRODUCTION

**FIFO Buffer** : As the name indicates when we try to read an element from FIFO it gives the element in the FIFO that entered first. FIFO buffer is used in video transmission and storage, matrix multiplication on multiple cores, serial communication, also it is one of the methods used in task scheduling by operating systems, also some times used in cache implementation.

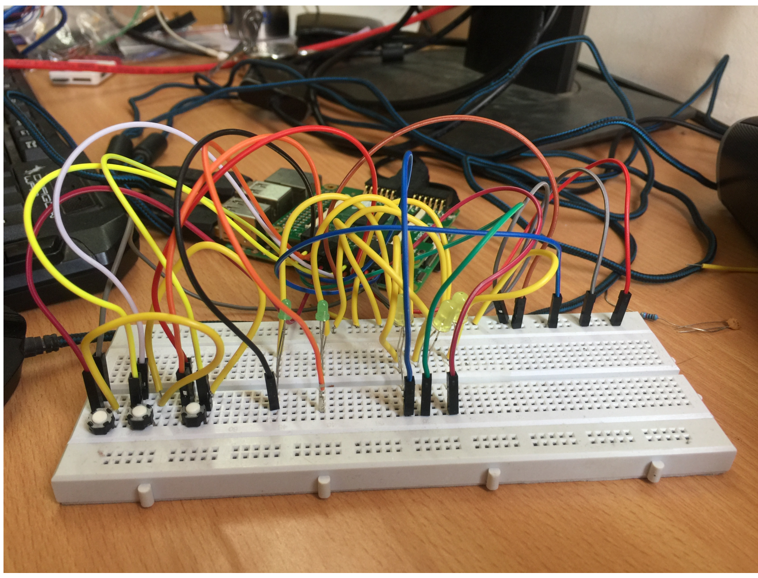


**icoBoard** is used as FPGA. It contains a Lattice FPGA with 8k LUT(Look Up Tables), 100MHz max clock, up to 8 MBit of SRAM and is programmable in Verilog by a complete open source FPGA toolchain. The icoTC(toolchain consisting of Yosys and ArachnePnR and icetools) can generate bitstream-files directly on the RaspberryPi. The icoBoard can also be operated standalone without RaspberrPi.IcoBoard consists up to 200 IO Pins or 20 PMOD modules can be connected to the icoBoard.



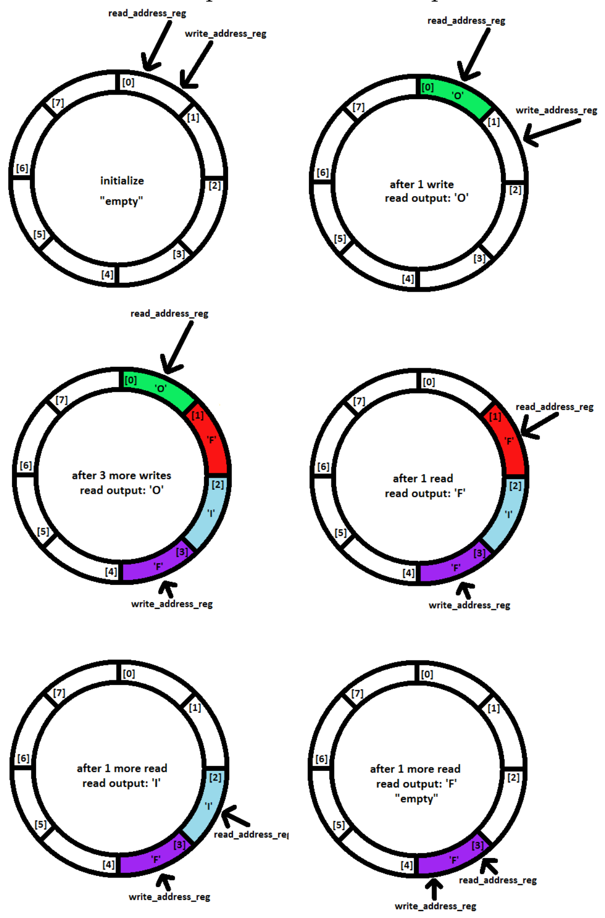
## 3 Components Required

1. icoBoard
2. RaspberryPi
3. Bread Board
4. LED's
5. push buttons

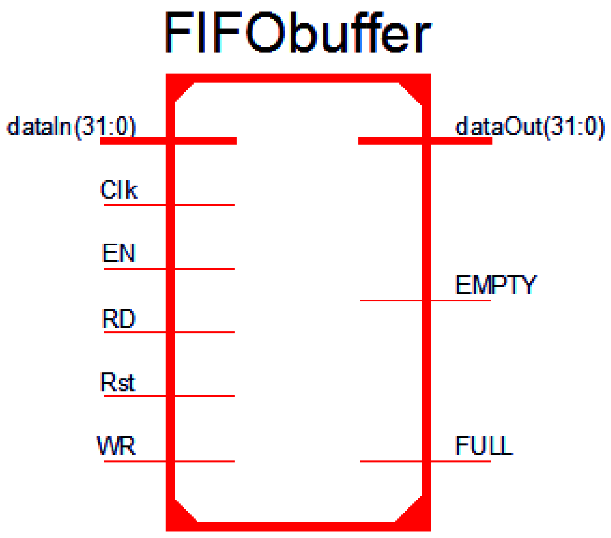


## 4 Implementation

**Software Implementation** : We write code in verilog. The algorithm works as we declare the size of each data entry(that is 3 bit or 8 bit) and the maximum number of data elements the buffer can accommodate. Depending on that we allocate memory to buffer. Then we use two registers(write register and read register) to store the point in buffer where to write or read which are initialized to 0. As already discussed we give read enable and write enable as inputs. So, whenever write enable is HIGH the write register is incremented by 1 and takes the input from input data pins. whenever read enable is HIGH read register is incremented by 1 and gives the data through output data pins. Coming to the empty and full flags. whenever write is enabled first we write empty flag as LOW, then check whether write register and read register have same values or not. If they are same then it imply FIFO buffer is full. whenever read is enabled first we write Full flag as LOW, then check whether write register and read register have same values or not. If they are same then it imply FIFO buffer is empty. This is the algorithm for FIFO. Then we need to write an .pcf(which should have same name as verilog file) file which contain pin configurations of icoBoard that is for all mentioned input and outputs in verilog code we assign io pins of icoBoard and write in .pcf file. Then we need to write a Makefile which runs our verilog code and .pcf files in icoTC to generates a bitstream files. This completes software implementation.



**Hardware Implementation** : Once software part is done hardware part is not a big deal. As mentioned earlier we assign pins to all outputs and inputs we connect outputs and inputs according to the pin configuration in .pcf file. We use raspberryPi as CPU so we connect icoBoard to raspberryPi and run the codes.



## 5 Conclusion

In this report we have seen how to implement FIFO on FPGA.We first discovered about how FIFO works, then about icoBoard and the tool chain required(icoTC). Then we have seen the algorithm to implement FIFO on our FPGA in software implementation. So, by this I conclude that the major thing in this project is to write the algorithm and write corresponding code in verilog. Once you are done with it is easier to do remaining parts of project.

## 6 References

1. http://simplefpga.blogspot.com/2012/12/fifofirst-in-first-out-buffer-in-verilog.html

2. https://embeddedthoughts.com/2016/07/13/fifo-buffer-using-block-ram-on-a-xilinx-spartan-3-fpga/

3. http://tlc.iith.ac.in/img/gvv_hemanth_icoboard.pdf