EE 314 Term Project

Spring 2022

FPGA Implementation of Simple Quality of Service (QoS) based Queuing

Naci Gülcü
2374981
METU EEE
Ankara, Turkey
naci.gulcu@metu.edu.tr

Muhammed Bahadır Demiryakan
2304426
METU EEE
Ankara, Turkey
e230442@metu.edu.tr

Hüseyin Necdet Devrekoğlu
2304434
METU EEE
Ankara, Turkey
e230443@metu.edu.tr

***Abstract*—This final report of the FPGA Implementation of Simple Quality of Service (QoS) based Queuing Project explains the working principles. This project has been written in Verilog HDL, and implemented on DE1-SoC FPGA board, and the output visualized in VGA interface on 640x480 resolution monitor.**

I. INTRODUCTION

We control the data traffic to ensure that overall system performance is optimized according to some criteria such as reliability or low latency requirements. We implement our project on the FPGA board. We generated input sequences, packets, by pressing the buttons, and used one button for "0" and one for "1". We send the inputs to the appropriate buffer which is determined by the first two bits of input. The buffers depth is 6. We minimized the packet drop. Then, in a first-in-first-out manner, we read the data by meeting the two important requirements, Latency and Reliability. Finally, we design a main screen using the VGA interface, at 640× 480 resolution. In this design, we represent the buffer arrays in different color with the data at their inside, and the total number of transmitted packets, received packets, and dropped packets of each buffer separately.

II. SUBUNITS

A. First In First Out

Firstly, the main idea here is shown as figure 1 below.

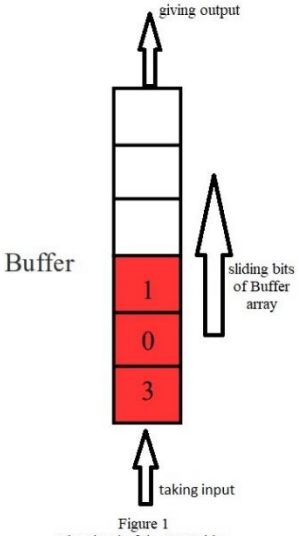

Figure 1
The visual of the FIFO idea

As shown here, while we are *Receiving Package*, the given input places to the bottom of the array and slides the older data to the upwards. While sliding, if there is a data at the top of the array, we lost that data, and we call this *Dropping Package*. While giving output, the algorithm that we used, checks the cells of the array from top cell to bottom cell and gives the first data it finds, which is the oldest data, first in data. By this logic, we give first input, oldest input as a first output of the buffer, and we call this data *Transmitting Package.*

B. Receiving Package

We give inputs by the start, 0 and 1 buttons of the FPGA board. In this part, for each buffer, we are using FIFO module that we created first, in order to receive and slide the packages. First, we push the start button of the FPGA board and start to give 4-bit binary input by 0 and 1 buttons of the FPGA. The 4-bit binary package that we give is in the form of ''B0B1B2B3''. There are 4 Buffers, and we named that buffers as *Buffer 1*, *Buffer 2*, *Buffer 3*, *Buffer 4*. We are deciding the buffer that receive the given data by the ''B0B1'' part of the data by using if statement. Buffer 1 takes the input if ''B0B1'' is equal to the ''00''. Buffer 2 takes the input if ''B0B1'' is equal to the ''01''. Buffer 3 takes the input if ''B0B1'' is equal to the ''10''. Buffer 4 takes the input if ''B0B1'' is equal to the ''11''. At the figure 2 below, the algorithm that we used is visualized.

After selecting the buffer, we give the binary 2-bit input which is ''B2B3'', then convert it to the decimal and write into the buffer array.

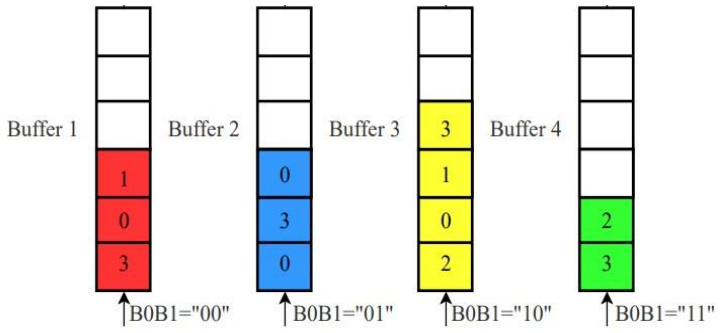

Figure 2

C. Transmitting Package

As receiving part, in transmitting part, we used FIFO module with requirements.

- Latency Requirement
- Reliability Requirement

Latency precedence for the buffers is given as Buffer 1 > Buffer 2 > Buffer 3 > Buffer 4.

Reliability precedence for the buffers is Buffer 4 > Buffer 3 > Buffer 2 > Buffer 1.

This means that the lowest packet drop rate is desired for Buffer 4 and also the lowest delay is desired for Buffer 1.

We achieved this requirement by checking the fullness and emptiness of the buffers with if else statement.

In order to reliability requirement, firstly, checks the fullness of the buffer 4, if it is full reads from buffer 4, if it is not full continues checking. Then checks the fullness of the buffer 3, if it is full reads from buffer 3, if it is not full continues checking. Then checks the fullness of the buffer 2, if it is full reads from buffer 2, if it is not full continues checking. Then checks the fullness of the buffer 1, if it is full reads from buffer 1, if it is not full continues to the latency requirement. After the reliability requirement, we realized that none of the buffers are full.

In order to latency requirement, first, checks the emptiness of the buffer 1, if there is any data at buffer 1 it reads from buffer 1, if it is empty continues checking. Then checks the emptiness of the buffer 2, if there is any data at buffer 2 it reads from buffer 2, if it is empty continues checking. Then checks the emptiness of the buffer 3, if there is any data at buffer 3 it reads from buffer 3, if it is empty continues checking. Then checks the emptiness of the buffer 4, if there is any data at buffer 4, it reads from buffer 4, else there is no data to read.

With these requirements, the minimum dropped package is desired from buffer 4 and the lowest delay is desired from buffer 1.

The visual of the requirements is shown below at figure 3.

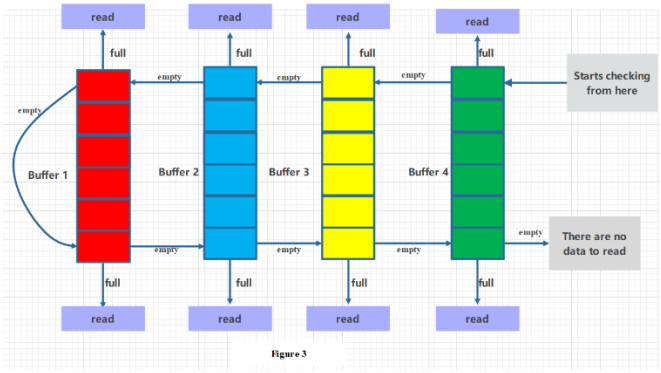

Figure 3

At every 3 seconds, a single package is read from selected buffer, and also it printed as output data. For each reading operation, the FIFO module updates the buffer array, deletes the read package, and updates the total transmitted package number.

First, we designed read enable for reading operation at every 3 seconds. We get read enable clock with 3 second period from FPGA boards clock inside with 50Mhz frequency. We created clock divider module for this process which counts at positive edge of clock until counter reaches the value 75 x 10^6.

We identify output data which is the package that read at every 3 seconds. Then we write this 4-bit output data to the VGA screen.

D. Design Unit

At this part of the project, we design the main screen using the VGA interface, at 640× 480 resolution. The screen represents the all-buffer arrays, and their data inside, also shows the total counted transmitted package, received package, and dropped package for each of the buffers separately.

In every pixel of VGA display (640x480), there are red, green, and blue lights within the power scale of 0 to 255. With changing the brightness of red, green and blue lights, we achieved desired colors at each pixel.

We design our screen as shown in the figure 4 below.

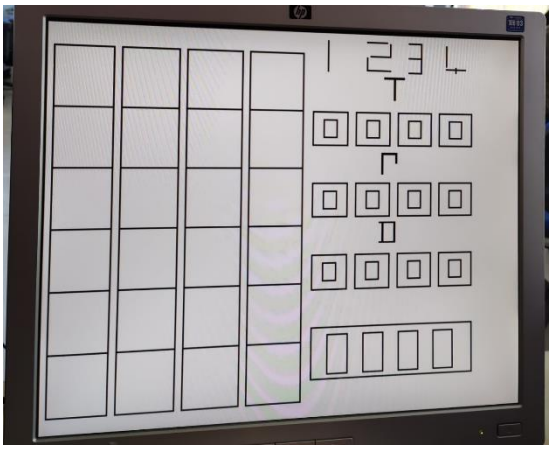
Figure 4

On the left side of our screen, we visualized each buffer arrays and their cells separately. We colored each cell when there are data in their inside, and also, we show the decimal data in their inside in seven segment display. Thanks to our VGA design, writing data and reading data processes can be observed continuously.

On the right side of our screen, we gave the number of transmitted, received and dropped packages for each buffer. And also, we gave the data output which is read at each 3 seconds continuously on the bottom of this side.

On the figures below, you can see the working systems photos step by step.

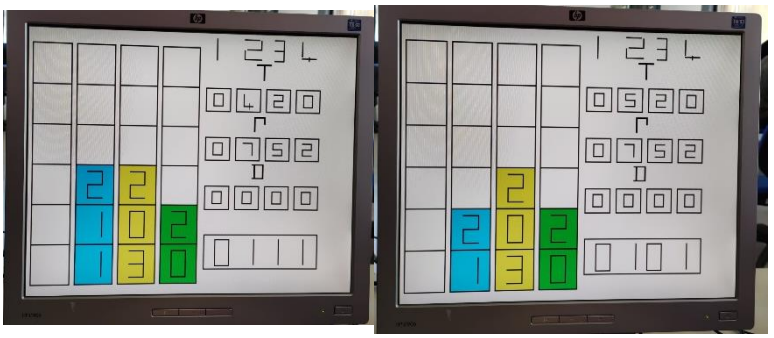
Figure 5

Figure 6

III. CONCLUSION

In this project, we develop Simple Quality of Service (QoS) based Queuing and its implementation on DE1-SoC FPGA board. We learned exercised and improved our digital electronics, logic design, using Intel Quartus software, Verilog HDL coding skills, using FPGA, also driving VGA interface and design by Verilog coding. We made research about First in First out idea and the importance of the queuing data. We faced some errors and problems during the project and managed to solve these and improved our algorithms. Finally, we reached good results and achieved the project requirements. To sum up, this was not a just a project for us, we enjoyed at all.