EE314-Digital Circuits Laboratory Final Report

1st Deniz Karakay
Electrical and Electronics Engineering
Middle East Technical University
Ankara, Turkey
e244330@metu.edu.tr

2nd Ömer Özer
Electrical and Electronics Engineering
Middle East Technical University
Ankara,Turkey
e237554@metu.edu.tr

3rd Mustafa Yılmaz
Electrical and Electronics Engineering
Middle East Technical University
Ankara,Turkey
e230574@metu.edu.tr

***Abstract*— The process we used to develop a Simple Quality of Service (QoS) based Queuing project for the EE314 Digital Electronics Laboratory course is detailed in this report. For our system to run, we created a Verilog solution and utilized VGA to demonstrate how this solution operates. We provide many Quartus simulation results that we have used to explain the theoretical aspects of the model we created. To further show the consistency of our approach, we compare theoretic and real-world results.**

***Keywords*— Verilog, FPGA, VGA, waveform, input, output, buffer, data, read**

I. Introduction

This project aims to create a Simple Quality of Service (QoS) based on Queuing. We are going to transmit data between input and output. Our data has four different types based on their first two-bit. Thus, we have four separate buffers. Inputs will be manually entered via FPGA and routed to the four different buffers. The inputs will be sent to the buffers according to the first two bits of data. The decimal form of the third and fourth bit is going to fill the buffers. Buffers are displayed with different colors, and the buffer depth is six packets. Whenever a new packet is received at an already full buffer, the oldest data is erased, and the newcomer is put to the order after one register shift of the remaining data. After all these processes, we will show the diagram of buffers and statistics, which are the number of transmitted packets transferred as output data, the number of received packets from FPGA buttons, and the number of dropped packets of the system on the screen using VGA.

II. Literature Research

In this phase, we tried to understand the problem correctly. Since our concern consists of multiple parts, we divide our project into these parts and try to find solutions to those problems. Until the buffer is half-filled, we choose some parameters which enable us to provide latency requirements. After the buffer is filled more than half, we choose another set of parameters to provide reliability requirements. We used Verilog codes to solve the problem and searched for more effective and less complicated codes. After providing all requirements, we need to show a buffer diagram and statistics on a screen. Thus, VGA is used for the user interface. More details are explained in the report.

III. Problem

In this project we need to transfer the data by using four different routes. After doing that we need to show the statistics about transfer by using VGA.

IV. Solution

A. *Input Mechanism*

While receiving the inputs, 0 or 1 input will be given over FPGA via buttons. These 0s or 1s will be memorized after every 4 bits and will appear in the input section on the screen. These 4-bit numbers will be stored at buffers after being classified by their first 2-bit. The example situation can be seen in Figure 1.

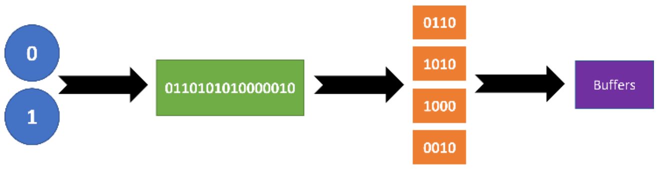


Fig. 1. Input Schematic of System Example

B. *Store Mechanism*

Each input received from FPGA must be placed in buffers according to a particular rule. Each input consists of 4 bits. The first 2 bits indicate in which buffer the data should be stored. The last two bits indicate the data to be stored. The first 2 bits of the input create four different variations in total. Four different buffers are used the store these four different variations. If the first 2 bits of the incoming data are "00", it is stored in the first buffer; if "01", it is stored in the second buffer; if "10", it is stored in the third buffer; if "11", it is stored in the fourth buffer. The example situation can be seen in Figure 2.

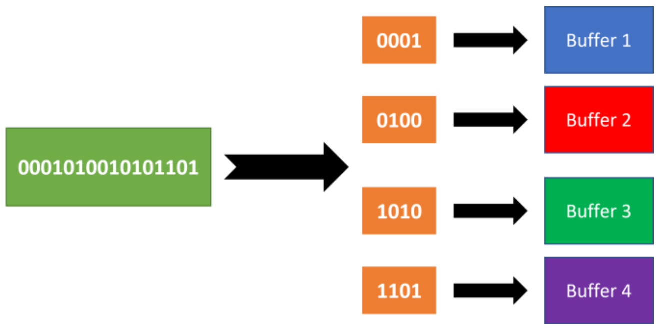


Fig. 2. Buffer Schematic of System Example

C. *Drop Mechanism*

If a full buffer receives a new packet, the oldest data is discarded (packet dropping), and the new packet is put into the queue after the remaining data has been shifted by one register. The example situation can be seen in Figure 3.

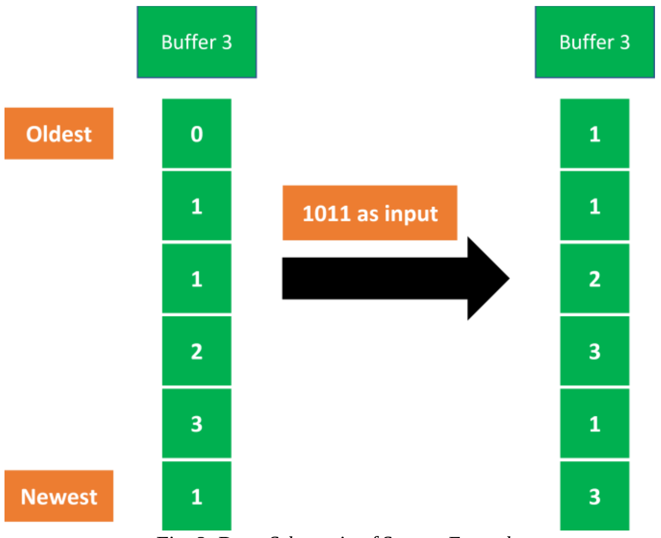
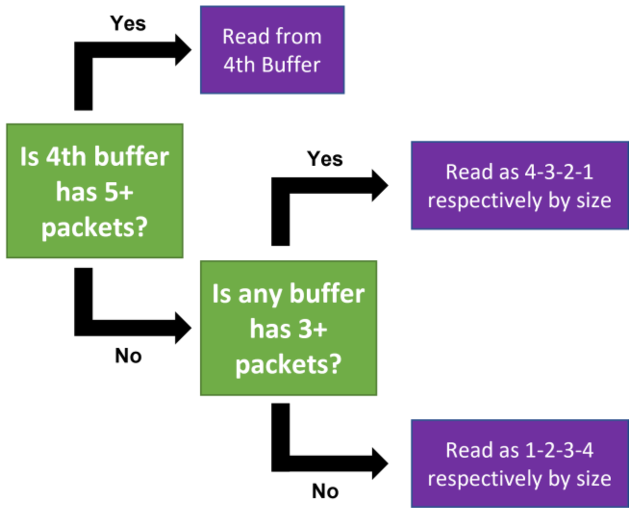


Fig. 3. Drop Schematic of System Example

D. Read Mechanism

We read 2-bit data from the system every 3 seconds. The oldest data is read first, in a first-in-first-out manner. After any data is read, it is deleted from the buffer. After that, all the data remaining in the buffer is shifted by one register. We decide which data will be read according to Latency Requirement and Reliability Requirement. We wrote an algorithm in Verilog for this process. If there are more than 3 data in any buffer, the system prioritizes reading according to the Reliability Requirement. The read priority goes as buffer 4 > buffer 3 > buffer 2 > buffer 1. In other words, priority is in the fourth buffer when there is a possibility of data dropping. If all buffers have three or fewer data, the system prioritizes reading according to Latency Requirement. The read priority goes as buffer 1 > buffer 2 > buffer 3 > buffer 4. In this way, the lowest delay belongs to the first buffer. The reading algorithm of our system can be seen in Figure 4.



Yes
Read from 4th Buffer
Is 4th buffer has 5+ packets?
Yes
Read as 4-3-2-1 respectively by size
No
Is any buffer has 3+ packets?
Read as 1-2-3-4 respectively by size
No

Fig. 4. Reading Algorithm of System

E. User Interface

For User Interface, we utilized VGA as known as Video Graphic Array port of the FPGA board. For our display, the refresh rate is fixed at 60Hz that is the typical for used in monitors. In order to accomplish this refresh rate on a resolution with 640x480 pixel, the calculation for pixel rate is done as follows:

Pixel Rate = (All Vertical Pixels * All Horizontal Lines *Number of screens / second) = 800 * 525 * 60 = 25 MHz

Therefore, we need to have 25MHz clock to program and display VGA output whereas we have 50 MHz clock on FPGA board. In order to solve this problem, we used mod-2 counter to generate 25 MHz VGA clock signal. After obtaining required clock signal, we implemented our mod-800 and mod-525 counter to enable VGA signal. We used *pos_H* for horizontal axis, *pos_V* for vertical axis. These values represent the position of each pixel within our 640x480 screen. We also checked the manual of our FPGA board, DE1-SOC manual, to learn the required pins for VGA, and its colors. [1]

Our FPGA board allows us to give color values for each pixel in 24-bit system which means 8 bits for red, 8 bits for blue and 8 bits for green. However, as you can see from the demonstration photos in *Fig 4-8*, we used multiple images for numbers, texts, buffers etc. Adding more images will cause longer compilation time. Therefore, we used 8 bits for colorizing the VGA, 3 bits for red, 3 bits for green and 2 bits for blue.

We utilized MATLAB for generating images on VGA display. In order to achieve that, we wrote a MATLAB script that reads the color values of each pixel in the image and converts it to 8 bit color value for VGA in hexadecimal form. We used *readmemh* function in our Verilog file to read those color values in memory in hexadecimal form. After obtaining those color data, we used the *pos_H* and *pos_V* values to show our images on VGA display.

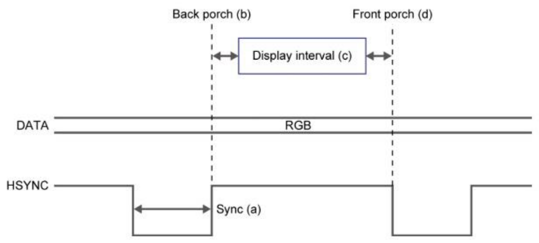


Fig. 5. VGA horizontal timing specification

V. Demonstration Photos

A. *Initial State*

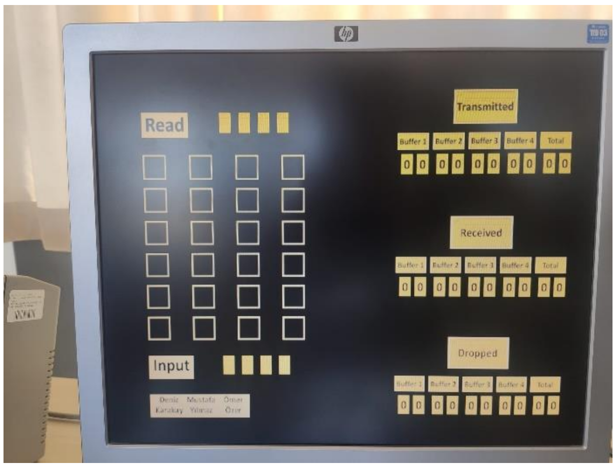


Fig. 6. Initial State of the VGA

B. *Read and Drop Mechanism*

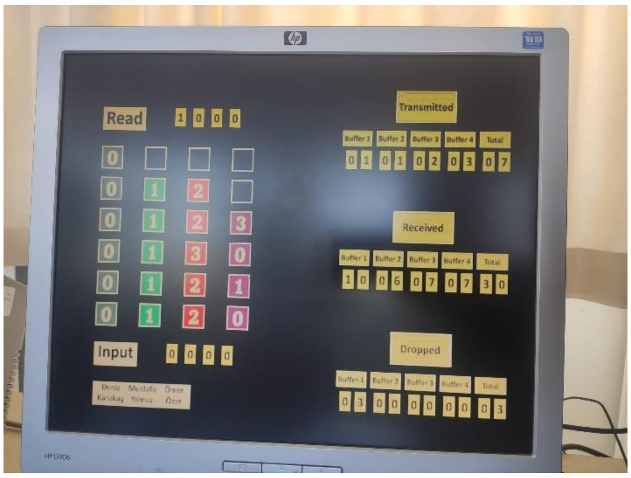


Fig. 7. A photo from the middle of the process of example 1

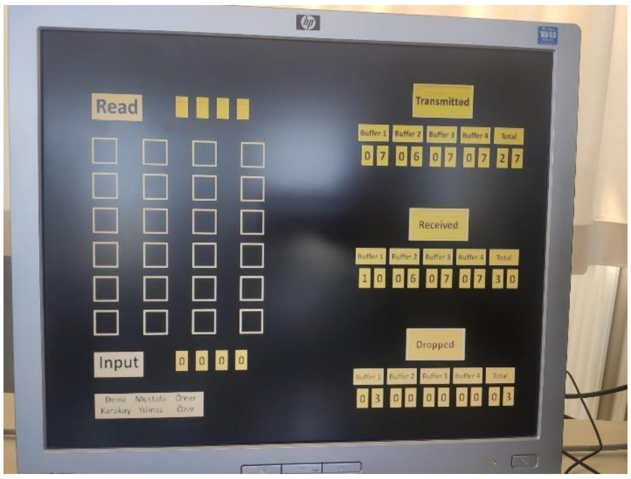


Fig. 8. Final Situation of example 1

This is the first example of read and drop mechanism demonstration. Here you can see in Figure 7 that some inputs have been given to the system and some of them were already dropped and some of them were read. 30 inputs are given in total. 7 of them have been read. 3 of them are dropped and the remaining 20 inputs are stored in buffers. You can see the final situation in Figure 8. All the remaining inputs are read.

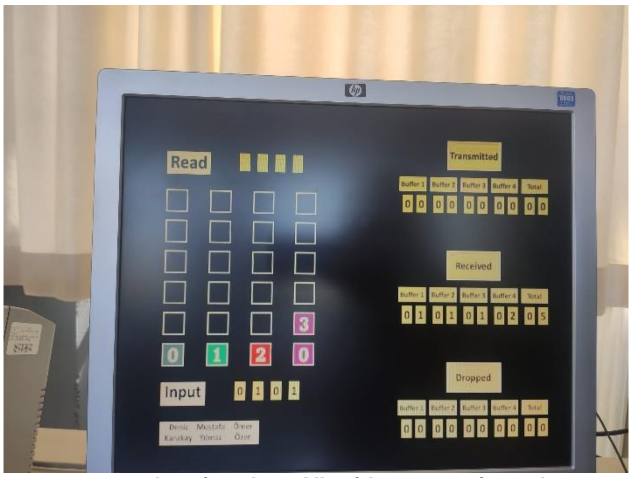


Fig. 9. A photo from the middle of the process of example 2

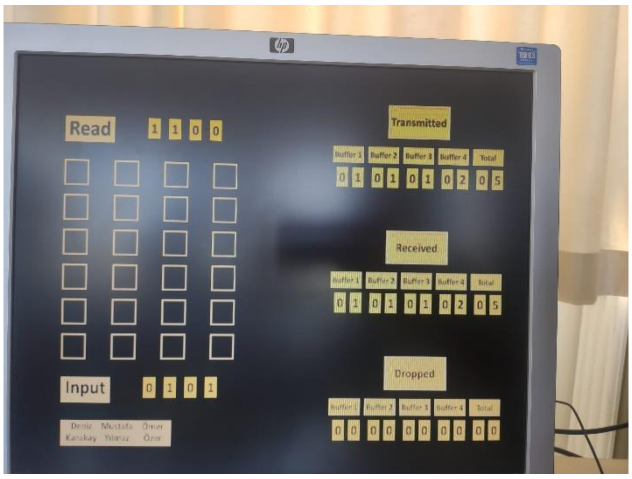


Fig. 10. Final Situation of example 2

In this example, we have given 5 inputs to the system as you can see in the Figure 9. All of them are read and transmitted. Final status can be seen in Figure 10.

C. *State Diagram*

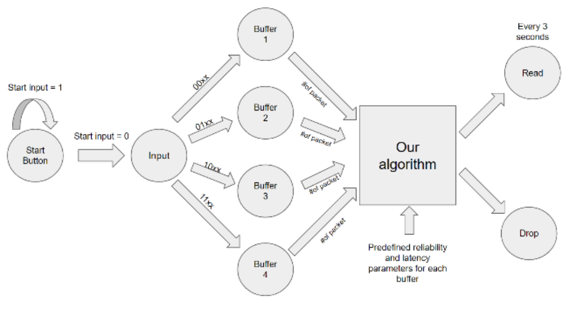


Fig. 11. State Diagram

Here you can see our state diagram in Figure 11. At the beginning, if we start the button, input process begins. According to the first two bits, we select at which buffer we will store the data. After that, our algorithm decides which buffer we should read data from.

VI. CONCLUSION

Creating a Simple Quality of Service (QoS) based on Queuing is the goal of this project. Data transmission between the input and output will be done. Based on the first two bits of our data, there are four possible kinds. There are four distinct buffers as a result. The FPGA will accept manual inputs and route them to the four separate buffers. The inputs will be transferred to the buffers following the first two bits of the data. The third and fourth bits in the decimal form will fill the buffers. The buffer depth is six packets, and the buffers are presented in various colors. The oldest value in a buffer that is already full is permanently deleted when a new packet arrives, and the newcomer is sorted after one register shift of the old data. After completing all these steps, we will use VGA to display a buffer diagram and statistics, including the number of packets transmitted and used as output data, the number of packets received from FPGA buttons, and the number of packets that the system dropped.

VII. REFERENCES

[1] F. www.fpgakey.com, "A VGA Interface in Verilog - Design Recipes for FPGAs Using Verilog and VHDL - FPGAkey", Fpgakey.com, 2022. [Online]. Available: https://www.fpgakey.com/tutorial/section892. [Accessed: 02- Jul- 2022].