EE 314 DIGITAL CIRCUITS LABORATORY 2022-2023 SPRING TERM PROJECT REPORT

FPGA IMPLEMANTATION OF A 2D STRATEGY GAME

Ahmet Caner Akar
Electrical and Electronics Engineering Department
Middle East Technical University
Ankara, Turkey
e244228@metu.edu.tr

Osama Awad
Electrical and Electronics Engineering Department
Middle East Technical University
Ankara, Turkey
e248849@metu.edu.tr

İsmail Enes Bülbül
Electrical and Electronics Engineering Department
Middle East Technical University
Ankara, Turkey
e244263@metu.edu.tr

***Abstract*—This document is about the end-term project of EE314 Digital Circuits Laboratory, implementation of a 2D strategy game by using FPGA.**

***Index Terms*—FPGA, Verilog HDL, VGA driver, button debouncing, state-machine**

I. Introduction

II. Project Overview

A. VGA Module

B. Button Debouncing and Edge Detector

In this project, we need to get three inputs, **logic 1**, **logic 0** and **activity**, from the user by using push buttons on the FPGA. However, due to mechanical and physical issues, pushbuttons often generate noisy signals called dirty bounces and these bounces preventing us to properly trigger the program. Thus, to eliminate these undesirable effects, we used *debouncer_delayed* module that makes a noisy pushbutton input signal to ideal input case.

The working principle of the *debouncer_delayed* is quite simple. When a button is pressed the timer is going to count the elapsed time up to a predefined threshold parameter. If the timer hits the threshold value, the program concludes that the button reaches its steady-state and it has been pressed. Similarly, when the button is released and the steady state is reached, the program concludes that the button has been released.

After debouncing button input signal, we designed another module called *edge_detector* to detect the negative and positive edges of the debounced input so that we will use the edge signals directly as button signals in the game controller.

The *button* module contains both *debouncer_delayed* and *edge_detector* module so that the hierarchical design principle is followed throughout the project. Also, both of these modules, are written like a state machine to make it easier to implement condition based and flexible code. The waveform simulation result of the *button* module in Quartus II is given in Figure 1, below. Besides these, in the *button_top* module each button is defined separately in a hierarchical manner by using *button* module.

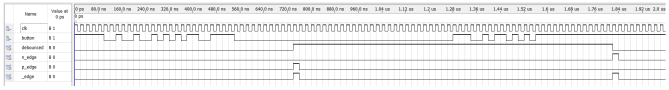

Fig. 1. Button module, waveform simulation result

C. Game Controller

Example citation [1]

III. Conclusion

References

[1] "Butterworth filter," Feb 2019.