Christopher Bero
6-2-2015
CPE 324

# Lab Report #10

# Single Button Texter

## Introduction

In this lab, we investigate a supplied lab and solution. For the purposes of comparing FPGA and ISP systems, the lab's code base has been implemented as a Quartus project and as a C program for use in a NIOS II processor. This project more heavily investigates the trade off of hardware and software implementations that arise when constructing a product such as the single button texter.

## Theory and Analysis

This system uses a modified morse code based on a default time period, which is then used as the limiter for short and long button presses and pauses. In the hardware implementation, this duration is cascaded from the initial clock division; the software implementation instead relies on a similar clock division with the caveat of less control over the running processor.

Post synthesis analysis is included in the lab manual. Results are discussed in the, erm, “Results” section of this report.

## Procedures

Procedure was provided in the lab manual, “Laboratory Assignment #10, Single Button Texter”.

## Results

We can see from the synthesis report that in all returned fields the hardware implementation uses less resources. This further confirms what we have been deducing

throughout the semester: hardware specific language is going to result in a smaller package. Conversely, the C program for our ISP is much more easy to read, maintain, and share among authors as it includes fewer references to specific hardware. As such, the software is bit-banging all communication, which means that adjusting (question 3) the clock frequency may disrupt or disable the peripheral communications. Similarly (question 4), the LCD controller has hardware specific peripherals which allow communication to be offloaded, while the VGA connection is bit-banged and required hardware specific methods.

## Conclusion and Discussion (also questions)

Debugging the system would vary in difficulty based on situation; the C program will most easily yield to algorithm and method induced errors, while the Verilog code may provide better insight to issues which arise from improper hardware allocation (question 2). For designing a hybrid system, the reasonable approach should be to wield the hardware available as effectively as can be quickly implemented (question 5). As far as pivoting a project to either hardware of software, software (with the pitfalls reviewed by this lab) should be used for purposes where the base hardware is subject to be changed, and future maintainability is key (question 6).