**Hamming Code Implementation on FPGA**

Written By: Connor Martin

8/27/2022

Abstract: A report detailing the implementation and results of implementing error correcting code (hamming code) into an FPGA.

Contents

[Chapter 1: Project Overview 3](#_Toc112511290)

[Chapter 1.1: Project Goal 3](#_Toc112511291)

[Chapter 1.2: Design Overview 3](#_Toc112511292)

[Chapter 1.3: High Level Block Diagram 4](#_Toc112511293)

[Chapter 1.4: Design User Guide 5](#_Toc112511294)

[Chapter 1.5: Hardware/Software Used 5](#_Toc112511295)

[Chapter 2: Implementation 6](#_Toc112511296)

[Chapter 3: Results 6](#_Toc112511297)

[References 7](#_Toc112511298)

# Chapter 1: Project Overview

## Chapter 1.1: Project Goal

The goal of this project is to show the implementation of Hamming code into an FPGA design. The Hamming code will be correcting a self-inducing error in the system. Data will be sent to the FPGA (via UART), displayed on a seven-segment display, and sent back out of the FPGA (via UART) to the serial monitor on a computer. The error inducing module will disrupt the data bitstream before the data is sent back to the computer. A discrepancy between what is on the seven-segment display and what is displayed on the serial monitor indicates an error has been introduced into the bitstream. Hamming modules (encoder and decoder) will be used to correct the error. The Hamming modules and error injection module will have the capability of being toggled on/off to show the effectiveness of each system.

## Chapter 1.2: Design Overview

This project will be using the modules provided by the UART receiver/transmitter project by NANDLAND [1]. Those modules include the UART reveiver/transmimtter and control for the 7-segment display. The debounce module from the debounce project by NANDLAND will also be used to control the LED interface [2]. The UART project by NANDLAND [1] will serve as the base design for the project. I will write a module that will inject erorrs into the design. The Hamming encoder/decoder written by Varun Jindal [3] will be inserted into the design to correct the self-induced errors.

The UART receive/transmit project by NADNLAND was chosen to be the base design for implamenting Hamming code into an FPGA for two reasons. The first reason is that the project uses UART. UART transmits 8 bits of data, which is required for the input of the Hamming encoder. The second reason was that the information received from the computer is displayed onto both the serial monitor and the seven-segment display. Injecting an error into only one of those data bitstreams will allow the user to see a discrepency between what the FPGA reads and what the FPGA writes. This helps viualize an error. The high level block diagram of the UART receive/transmit project provided by NANDLAND can be found below in figure 1.

Diagram

Description automatically generated

Figure 1: UART Receive/Transmit High Level Block Diagram

## Chapter 1.3: High Level Block Diagram

This project’s high level block diagram can be found below in figure 2. It is very similar to the high-level block diagram of the UART receive/transmit project by NANDLAND. The only difference is the Hamming code and error injection modules have been inserted into the design.

Diagram

Description automatically generated

Figure 2: High Level Block Diagram

## Chapter 1.4: Design User Guide

Diagram

Description automatically generatedFigure 3 below serves as a user manual to the project. Pressing button 1 on the FPGA development board toggles the error injection module on/off. Pressing button 2 on the board toggles the Hamming code on/off. LED 1 and LED 2 are connected to button 1 and button 2 respectively. When LED 1 is high, the error injection module is active. When LED 2 is high, the Hamming code is activated.

Figure 3: Project User Guide

## Chapter 1.5: Hardware/Software Used

**Hardware:**

NANDLAND Go Board (FPGA development board) [4]

Computer [Dell G15]

Micro USB cable capable of data transmission

**Software:**

Lattice iCEcube2 (synthesize code) [5]

Diamond Programmer (program FPGA) [6]

Notepad++ (text editor) [7]

Tera Term (serial monitor) [8]

**Verilog Modules:**

Hamming encoder [3], Hamming decoder [3], UART Transmitter [1], UART receiver [1], Binary to 7-segment display converter [1], Debounce [2]

# Chapter 2: Implementation

## Chapter 2.1: Error Injection

The error injection module is used to simulate a one-bit error occurring in the data bitstream. This module ensures that the final data bit in the UART bitstream is a one. This simulated error occurs after the data bitstream has run through the Hamming encoder and before the Hamming decoder. The error injection module is set up to receive a 12-bit input, which is how many bits the data has once it has been encoded by the Hamming encoder. If the data has not been decoded, the module will accept the original 8-bit data input into the 11-bit input. When looking at the figure below, you can see that when a ‘b’ is pressed on the keyboard, a c will show up on the serial monitor when the error injection module is activated. This is because the final data bit in the data bitstream is switched to a one. A data bitstream that already has a one in the final bit is unaffected.

Text, letter

Description automatically generated

Figure 4: Error Injection Module

## Chapter 2.2: Hamming Encoder/Decoder

The Hamming encoder and decoder are located before and after the error injection module respectively. The Hamming error correcting code has the capability to correct a single bit error. This ensures that even though the final bit may have been switched to a one after the encoding, it is switched back to a zero after the decoding.

## Chapter 2.3: User Interface

The error injection module and Hamming modules have been coded to allow for human interface. This simply allows the user to toggle each system on/off. This helps show how the error injection works and Hamming correcting code works.

# Chapter 3: Results

The results from this experiment are as predicted. When neither the Hamming modules nor error injection module are active the device works as it was originally intended. Once the error injection module is activated, the final data bit in the data bitstream that is sent back to the computer is always a one. Once the Hamming code is activated, the data bitstream that is sent back to the computer is corrected to its original value. The device also works normally when only the Hamming modules are activatedb

# References

[1] UART Receiver/Transmitter: <https://nandland.com/project-8-uart-part-2-transmit-data-to-computer/>

[2] Debounce a Switch: <https://nandland.com/project-4-debounce-a-switch/>

[3] Developing Hamming code using Verilog HDL: <https://www.researchgate.net/publication/271444656_Developing_Hamming_Code_Using_Verilog_HDL>

[4] GoBoard by NANDLAND: <https://nandland.com/the-go-board/>

[5] Lattice iCEcube2: <https://www.latticesemi.com/iCEcube2>

[6] Diamond Programmer: <https://www.latticesemi.com/programmer>

[7] Notepad++: <https://notepad-plus-plus.org/>

[8] Tera Term: <http://www.teraterm.org/>