(7,4,3)Hamming Code on FPGA

Zohan B Philip
ee16tech11039@iith.ac.in

Sanjay M
ee16btech11034@iith.ac.in

April 27, 2019

Abstract

(7,4,3) Hamming Code is one among several error correcting codes. It encodes a 4 bit message onto a 7 bit codeword using a Generator Matrix. The Syndrome is calculated by multiplying the received codeword with the parity check matrix. The syndrome can be used to decode the incoming message. The hamming distance is three.

I. Components

1) Raspberry pi
2)Arduino UNO
3)12 Male to Male Jumper Wires
4)Icoboard(FPGA)

II. Circuit Connections

Icoboard is mounted on top of the Raspberry Pi and Arduino UNO is connected to the Raspberry Pi via a USB cable.
The following corresponding connections are made between arduino and Icoboard digital pins using the jumper wires.
2 - B6
3 - B3
4 - B5
5 - A5
6 - A2
7 - C3
8 - B4
9 - D8
10 - B7
11 - B9
12 - B10
13 - B11

III. Procedure

i. Input

First we need to generate 4 bit random messages and write them onto a file using python.

ii. Encoding

We must send the input message from the input file to the FPGA using python and Arduino, and retrieve the encoded 7 bit codeword from the FPGA using python and Arduino and write it onto another file.

iii. Noise

We must add noise to the encoded message and write it onto another file using python.

iv. Decoding

We must send the noisy message from the noise file to the FPGA using python and Arduino, and retrieve the decoded 4 bit message from the FPGA using python and Arduino and write it onto another file.

IV. Codes

i. input.py

Generates random integers from 0 to 15 and writes them onto the file, input.txt

ii. encode2.py

Reads the integers from input.txt and sends them to serial port.
It also reads a 7 bit integer from the serial (from encoder.ino) and writes it onto the file, encoded2.txt

iii. encoder.ino

Reads the 4 bit integers written on the serial port by encode2.py, splits them into 4 bits and send the 4 bits to the FPGA. It then receives 3 parity bits and combines all 7 bits into an integer and prints it on the serial port.

iv. hamm.v

Generates the parity bits according to (7,4,3)Hamming code. Its clock is triggered by encoder.ino

v. hamm.pcf

Specifies the FPGA pins assigned to each register for hamm.v

vi. noise2.py

Takes the 7 bit messages from encoded2.txt, adds AWGN noise (using OOK modulation scheme) to each bit, finds the received distorted 7 bit vector. It writes the received vectors onto the file, noisy2.txt

vii. decode2.py

Reads the 7 bit integers from noisy2.txt and sends them to serial port.
It then reads a 4 bit integer from the serial (from decoder.ino) and writes it onto the file, decoded2.txt

viii. decoder.ino

Reads the 7 bit integers written on the serial port by decode2.py, splits them into 7 bits and send the 7 bits to the FPGA. It then receives the 4 bits corresponding to the decoded message and combines the 4 bits into an integer and prints it on the serial port.

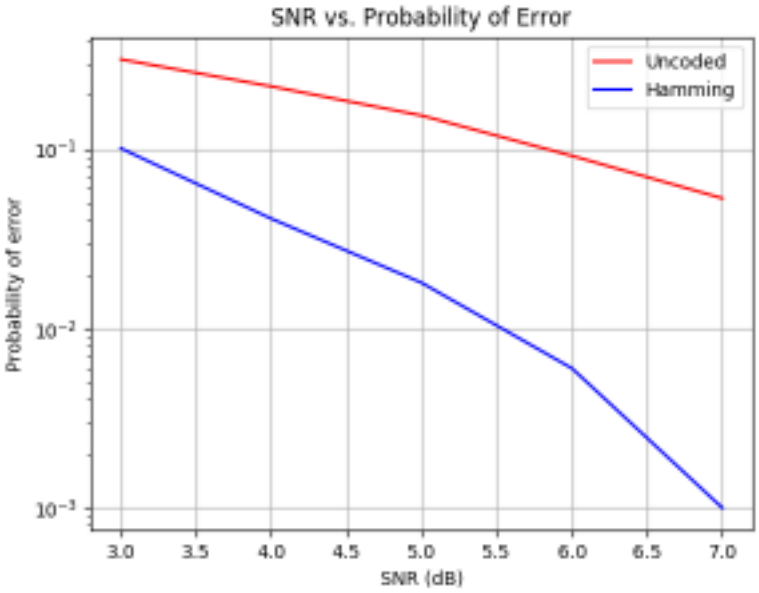
ix. hammdecode.v

It calculates the syndrome corresponding to the 7 bit codeword received from decoder.ino, finds the error and sends the decoded message to decoder.ino

x. hammdecode.pcf

Specifies the FPGA pins assigned to each register for hammdecode.v

V. Plot



By increasing the number of samples we can estimate the probability of error for a given SNR .The plots above are the plots of SNR vs probability of Error for Uncoded communication and Hamming code communication.