GSOC 2016 : Implementing an optimized and standardized channel code

Personal Information

Name : Tane Juth Tangu,
Level : Undergraduate
School : University Of Buea,Cameroon
Contact: tanejuth@gmail.com ,tel (+237)678431904,irc:tanerochris
Time : GMT +1
Language :English and French spoken and writtten.
Programming Language : c/c++,javascript,php

Introduction

When data is to be transmitted across a noisy channel,there is the possibility of errors being introduced into the transmitted data.Thus, there is need to identify such errors and correct them at the receiver end.This is known as forward error correction.This proposal is based on implementing a Forward Error Correction code as specified by some standards example Wimax,turbo just to name a few.And the code shall be optimized such that they shall be used in real life applications.I shall be implementing QC-LDPC(quasi-cyclic) codes as specified by the Wimax standard for FEC.An LDPC code is said to be quasi-cyclic if for any cyclic shift by *c* places the resulting word is also a codeword.Wimax standards specifies the parity check matrix as right shifted circulant matricsWe will see what is and what is not yet in the gnuradio gr-fec and see if there is any existing module that needs improvement and propose ideas to effect improvements.The proposal is better laid out as WH questions.

What is needed to implement the FEC based on Wimax Standards code?

We need a source block that provides the systematic bits,parity check matrix generator block this shall be used to generate parity check matrices basically used for QA test,we need an encoder block, a modulator block , a channel across which bytes are transmitted and a decoder block.

What is available in Gnuradio?

We will ignore the source and the sink blocks.There is a general LDPC encoder and decoder block gr-fec/lib, AWGN channel still in gr-fec/lib.There is also qpsk modulator and demodulator blocks.All these are necessary to successfully implement FEC.Due to the fact that the standard for which the FEC is implemented ,IEEE 802.16e ,and need for optimization,might lead to slight modifications on some already existing blocks for example the encoder,and FEC deployments as we are dealing with QC-LDPC codes.

What did mentors suggest?

Still to be filled and dearly needed.

What will be added or improved or optimized?

Optimizing Memory

Only none-zero value *i,j* positions of a sparse matrix will be stored,so as to maximize memory.Example if all bits of a parity check matrix were to be stored in an array will occupy N_z*z*8 bytes of memory,whereas if non-zero elements are stored they will occupy N_z*8 bytes of memory given that we store the non-zero positions of the base matrix in a vector P,and the corresponding permutation values in another vector S and address an element as address[i*z + j] = P[i]*z + (S[i] +j)mod z [1].The boost library has classes for manipulating sparse matrices.Including GNURadio's GSL library.

Encoder

Arbitrary bit-generation and correction algorithm is proposed as the encoding algorithm as it exploits the dual diagonal nature of QC-LDPC codes and has a reduced time complexity than the conventional efficient Richardson algorithm[2] .Matrix operations like scalar or vector multiplication will be done with the use of volk api..S bits shall be encoded simultaneously by vectorizing the process.

Decoder

The a Single Scan MIN-SUM algorithm will be used by the decoder block.It merges the horizontal scan ,check node update, and the vertical scan,variable node update, of the original min-sum algorithm into a single horizontal scan.[3].The algorihm is presented below.
Where x' decoded word,y = received word,Z_n=priori log likelihood ratio

procedure DECODE (Zn)
Initialization :

k = 0
for i =1: n
$Z_i^{(0)} = \ln(P(x_i=0/y_i)/P(x=1/y_i))$
end for
for i =1: n
for j=1: m
$L_{ji}^{(0)} = 0$ //using the boost matrix to initialize,will avoid this step
end for
end for

repeat
for j = 1 : m do
for i $\in$ B N_j do
$L_{ji}^{(k)} = \Pi_{w'} \mathrm{sgn}(Z_i^{(k-1)} - L_{ji'}^{(k-1)}) * \min_{w'} | Z_i^{(k-1)} - L_{ji'}^{(k-1)} |$
$Z_i^{(k)} \mathrel{+}= L_{ji'}^{(k)}$
end for

end for
for i = 1 : n do
$L_i^{(k)} = Z_i^{(0)} + \sum_v L_{j',i}^{(k)}$
If $L_i^{(k)} \leq 0$ then $x'_i = 1$
else $x'_i = 0$
end if If $k = k_{max}$ OR $\mathbf{H}(\mathbf{x'})^T = \mathbf{0}^T$
Finished
else k= k +1
until Finished
end procedure

Multiple codewords will be decoded simultaneously with the use of SIMD instructions.

QC-LDPC constructor

An implementation of the algebraic construction of QC-LDPC code by dispersion will be done.As these codes perforn well over an AWGN channel.[4]

Python Tests

QA(Quality Assurance) test codes shall be developed in python for the various blocks.This will be used to ensure that the encoder and decoder are error free,for quality software.

Provide encoder and decoder as FEC API variables

The developed encoder and decoder are provided as fec api variables.

Documentation

A documentation of the deployments and variables shall be done.

Why do people say that Polar codes are good?

TIMELINE

Below is a proposed timeline for the summer,I will be coding from the period 22April to 20 June ,will be delivering 36 hours per week ,4 hours per day and covering up on Saturdays and Sundays.From 9 July to the end of GSoC ,I will developing 40 hours per week.Pushes will be done on Sundays.There are period that I will not be coding 25 April - 30 April where I will be writing CA's and 21 June - 08 July where I will be writing Semester Exam.I will love to say that,the University calender might change,in case that is the case I will inform my mentor,and shall code in the aforementioned period.I will be coding mostly in the period of 8PM GMT.

Period	Week Count	Task	Activities	Deliverables
25March - 22 April	4	Getting familiar with GNURadio code base.	-Resolving An issue -Learn python -Learn SIMD -Learn Volk	-resolved issue
22April -22 May	4	Community bonding period	-Discuss methods of optimizing encoder and decoder algorithms -implementing QC-LDPC construction algorithm	-QC-LDPC constructor code -Any assigned task
23 May - 5June	2	Encoder	-encoder deployments -encoder variables -optimization -unit test	-encoder code
6June - 19 June	2	Decoder	-decoder deployments -decoder variables -optimization -unit tests	-decoder code
9 July - 16 July	2	Python tests	-encoder deployments and variables test code -decoder deployments and Variables test code	-test codes
17 July - 30 July	2	Integration Test	-python hierarchical block to test entire FEC -combining modulator blocks,channel ,encoder and decoder	-hierarchical block
31 July - 14 August	2	-FEC API variables -GRC	-Provide encoder and decoder as FEC API -create .grc files for the encoder and decoder	-grc blocks
15July- 23 August	1	Documentation and	-Document code	-code

		proper comments	on doxygen -Document grc blocks -Improve comments	documentation

Why contribute to GNURadio?

My interest in GNURadio started when I was looking for the perfect software to realize my final year project.The project about providing a mobile application that will enable users on a LAN,like a campus, to make free calls or messaging ,without going through the web or GSM network.I came accross software defined radio which later led me to GNURadio. So contributing to GNURadio is an opportunity for me to master the core tool I need for my final year project ,and why not make it available to the community.

Experience

I am specializing in Software Engineering at my University,successfully completed a course on DSP.Well since this is not a summer Of research,I've for the past two weeks done my Research on LDPC,their construction,encoding,decoding and different algorithms and their optimization.I am not knew to open source and neither to GSoC program as an applicant.I have some few codes on github,but I have never contributed to an organization yet.

The three strike rules

References

[1] Gabriel Falcao et al "LDPC Decoders for for the Wimax(IEEE 802.16e) based on multicore architectures"

[2]Arbitrary Bit-Generationand Correction Technique for Encoding QC-LDPC Codes with Dual-Diagonal Parity Structure.Chanho Yoon, Eunyoung Choi, Minho Cheong and Sok-kyu Lee

[3]Single-Scan Min-Sum Algorithms for Fast Decoding of LDPC Codes .Xiaofei Huang

[4]Y. Y. Tai, L. Lan, L. Zeng, S. Lin and K.A. S. Abdel-Ghaffar, "Algebraic Construction of Quasi-Cyclic LDPC Codes for the AWGNand Erasure Channels," IEEE Transaction on Communications, vol. 54, no. 10, pp. 1765-1774, October 2006.