GSOC 2016 : Implementing an optimized and standardized channel code

Tane Juth Tangu

University Of Buea,Cameroon,

Email : tanejuth@gmail.com

March 2016

1. Personal Information

Level : Undergraduate

Tel (+237)678431904,irc: tanerochris

Town : Buea

Time zone: West Africa Time (WAT) +0100 UTC

Language :English and French spoken and written.

My GSoC work is not in connection with my University and I will not be receiving any credit for it.

2. Background

I am a third year student at the University Of Buea, undergoing studies for a B.Eng in Computer Engineering.This is the first time I will be contributing to an open source organization ,an I believe starting with GNURadio will be an awesome opportunity for me to join the open source community.Not only that, also I intend to use GNURadio as a tool, in developing an application based on providing low cost communication in a LAN, for my final year project.So getting involved ,with GNURadio, is kind of being on the right footing .

I have been working for a while on LDPC and QC-LDPC codes in particular as relating to the project,my understanding of the subject matter was first eased by the fact that I had completed and validated a DSP course in school so I had some idea of coding theory.I do code in c ,c++ , node js and php.

I have read and acknowledged the three strike rules.And before introduction I will love to say that Polar Codes are the coolest codes ever.

3. 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 ,especially in systems where the correctness of the received data is critical,like in video conferencing applications.To correct these errors ,redundant bits,error correcting codes, are added to the transmitted bits prior to transmission and at the receiver end any errors in the transmitted data is identified and corrected without the need for re-transmission ,in the event that data was corrupted on transmission via a noisy channel.This process is know as Forward Error Correction.There are error correcting codes like hamming codes,LDPC codes,RS codes just to name a few that are used to correct transmitted data.This proposal is about implementing Forward Error Correction based on the LDPC family of error correcting codes, as specified by some standards example IEEE 802.16e which I have chosen to work on .There are blocks involved in the transmission of data ,these are a basically an encoder, demodulator,modulator,noisy channel and a decoder .Generally, these blocks are implemented in hardware and since we are dealing with software defined radio,there is need that their software implementation be able able to work at the same speed and efficiency to be used in real life applications.So there is need to stress on optimization of these blocks.

4. LDPC(Low Density Parity Check) Codes

LDPC codes are error correcting codes that consist of a sparse matrix,that is a matrix where number of 1's in any column and row is far far less than the number of columns and rows.GNURadio already has and implementation of a general encoder and decoder for these codes in gr-fec.The IEEE 802.16e specifies a particular kind of LDPC codes,QC-LDPC(Quasy-cyclic LDPC) codes for FEC.This matrix has a parity check matrix with a dual diagonal identity matrix structure,and where each entry is a circular right permutation of the Identity matrix.The main concern will be focused on addressing optimization of these codes from their construction,encoding and decoding.

The general problem faced with LDPC is their memory utilization and lots of random access.Storing only the none zero entries saves us some memory ,for example ,if all bits of a parity check matrix were to be stored in an array it will occupy N_z*z*8 bytes of memory,where N_z is the length of the matrix and z is the

length of the sub-matrix, 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 ,we can address elements 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.

It is important that the algorithms used for encoding and decoding be optimized as mentioned earlier.The encoding algorithm used by the GNURadio's is the Richardson and Urbanke [2]which encodes near linear time($O(n+g^2)$) if g,the gap equals 0,the complexity becomes linear.Chanho Yoon et al propose an arbitrary bit-generation and correction encoder algorithm that exploits the Dual diagonal structure of the QC-LDPC code ,has a low complexity and runs faster ,and is suited for the IEEE 802.16e LDPC codes,than the Richardson and Urbanke algorithm[3].For decoding Xiaofei Huang proposes a single scan min-sum algorithm for fast decoding. It reduces memory usage because it only needs the addressing from check nodes to variable nodes while the original min-sum algorithm requires that addressing plus the addressing from variable nodes to check nodes. It has shown that ,the single-scan min-sum algorithm, it is more than twice as fast as the original min-sum algorithm[4].

There will be lots of matrix operations like addition and multiplication going on,and also running the encoding and decoding algorithms on multiple data simultaneously.This is to achieve speed.There is the volk library in GNURadio that will be used for SIMD optimization [5]of the encoder and decoder algorithms and any other process requiring such approach.This will entail designing the above algorithms so as to make maximum use of VOLK.

5. Deliverables

Encoder

Encoder deployments and variables will be developed.

Decoder

Presented below is the Single-Scan Min-Sum algorithm.

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

$L_{ji}^{(k)} = \Pi_{w'} \text{sgn}(Z_i^{(k-1)} - L_{ji'}^{(k-1)}) * \min_{w'} | Z_i^{(k-1)} - L_{ji'}^{(k-1)} |$

$Z_i^{(k)} \mathrel{+}= L_{ji'}^{(k)}$

end for

end for

for i = 1 : n do

$L_i^{(k)} = Z_i^{(0)} + \Sigma_v L_{j'i}^{(k)}$

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.

Decoder deployments and variables shall be codQA(Quality Assurance) test codes shall be developed in python for the various blocks.This will be used to ed.

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.[6]

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.

LISCENSE

All Code developed will be under the GPLv3 license.

6. TIMELINE

Below is a proposed timeline for the summer,I will code from the period 22nd of April to the 20th of June ,where I will be coding 36 hours per week ,4 hours per day and covering up on Saturdays and Sundays.From 9th of July to the end of GSoC ,I will be coding for 40 hours per week.Pushes will be done on Sundays.The period from 21st June till the 08th July ,I will not be coding,reason being that I will be writing Semester Exams.I will love to say that,the University calender might change, in that case I will inform my mentor,and shall code in the aforementioned period.I will be coding mostly from the period of 8PM GMT.

Period	Week Count	Task	Activities	Deliverable(s)
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	-Discuss methods	-QC-LDPC

		bonding period	of optimizing encoder and decoder algorithms -implementing QC-LDPC construction algorithm	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 files	-Provide encoder and decoder as FEC API -create .grc files for the encoder and decoder	-grc blocks
15July- 23 August	1	Documentation and proper comments	-Document code on doxygen -Document grc blocks -Improve comments	-code documentation

References

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

[2] Modern Coding Theory BY T. R I C H A R D S O N A N D R . U R B A N K E

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

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

[5] Thomas W. Rondeau ,Nicholas McCarthy ,Timothy O'Shea "SIMD Programming in GNU Radio: Maintainable and User-Friendly Algorithm Optimization with VOLK"

[6]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.