Google Summer of Code 2014 : Performance Optimization with VOLK

Abhishek Bhowmick
abhowmick22@gmail.com

Potential Mentor: Nathan West

March 15, 2014

Contents

1 Introduction

GNU Radio [1] is a software development toolkit that can be used to implement software defined radios using commodity hardware. The signal processing blocks it provides are mostly run on general purpose processors, and performance is a key metric for the implementations on various architectures. With this in mind, GNU Radio includes VOLK, which is a library of kernels that utilize vector SIMD extensions to popular architectures (SSE/AVX for x86, FMA for AMD64, Neon for ARM). Current VOLK kernels target stock operations such as vector multiply, conjugate, dot product etc, but various benchmarking studies have shown that more complex kernels deliver better runtime performance [2].

2 Proposal

The objectives of the project are to enhance the functionality/performance of the VOLK library and develop blocks for better OFDM performance. This will be achieved in two stages. First, AVX implementations will be developed for the current VOLK kernels, and turbo encoder/decoder blocks from OpenAirInterfaces [3] will be ported to VOLK. Next, new kernels will be developed to accelerate the in-tree receiver OFDM implementation, specifically targeting operations such as preamble detection, symbol timing and fine frequency offset correction, eventually leading to a complete OFDM frame detection kernel. New kernels will also be identified based on profiling studies, to accelerate performance bottlenecks. In-tree OFDM, DVB-T, ATSC and Fecapi were identified as potential candidates to speed up with new kernels. As a modification to the original idea, better performance of OFDM will be targeted through signal processing techniques.

3 Profiling and New Kernels

Some prelimiary profiling was done for the example OFDM implementation to identify performance bottlenecks. Figure 1 identifies the most computationally intense blocks in OFDM Rx through a Control Flow Graph.

Oprofile [4] was used to further identify the C++ routines within these blocks that were processor heavy. For instance, the *fir_filter_ccf* and *fir_filter_ccc* blocks spent a lot of time in their respective filter() methods, that were already seen to have some vectorization with the *volk_32fc_32f/x2_dot_prod_32fc_a* kernels. AVX support and better vectorization can be used to target these regions. Similarly, the *work()* methods of *fractional_resampler_cc*, *additive_scrambler_bb* and *ofdm_frame_equalizer_bb* may also be accelerated through vectorization. Detailed profiling will be carried out in other modules to identify bottlenecks that can be sped up with new kernels.

4 OFDM Receiver

Orthogonal Frequency Division Multiplexing is a method for encoding digital data on multiple carrier frequencies that are orthogonal to each other. In an OFDM receiver, the signals received from channel are down-converted, filtered, sampled

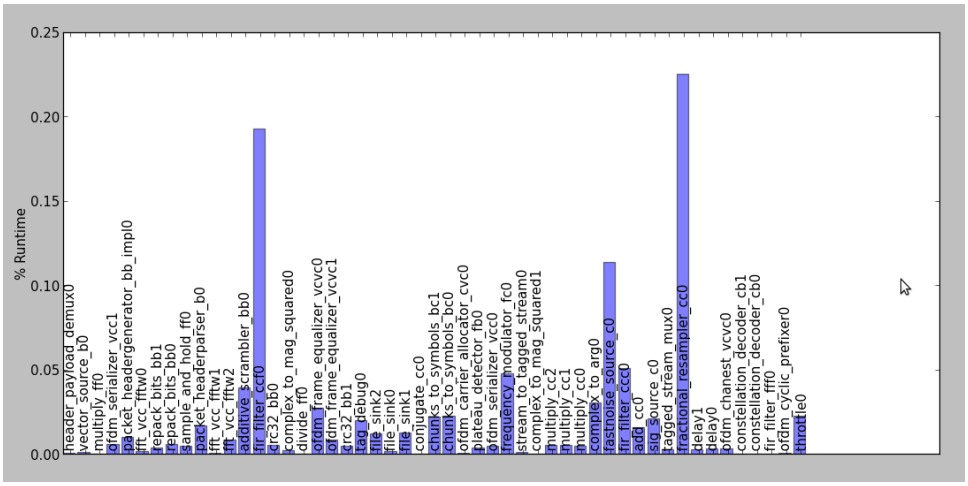

Figure 1: Percentage runtime of different blocks in OFDM RX example

and digitised. An FFT block converts these samples back into the frequency domain. The parallel streams from the FFT are then decoded into symbols using symbol detectors and serialized to get back the original data stream.

4.1 Frame synchronization

The synchronization block is needed to detect the beginning of the symbol and correct fine carrier frequency offset. The benchmark OFDM_rx implements the Schmidl Cox Frequency/Timing Synchronization [5], which relies on a two-symbol training sequence, placed at the beginning of a frame. In the current implementation, symbol timing is done in the *ofdm_sync_sc_cb* block while coarse frequency offset calculation is done in *chanest_vcvc*. As both these operations are similar in nature, a new cross-correlation kernel will be developed to accelerate them. Another promising direction is to implement Awoseyila's method [6] that uses only one training symbol to achieve time-frequency synchronization entirely in the time domain, that gives it the benefits of low complexity and fast convergence. As all processing is done in the time domain, we can combine the split frequency synchronization in the current OFDM implementation into a single frame synchronization kernel.

4.2 Channel Estimation

The *ofdm_chanest_vcvc* block does the channel estimation and coarse frequency offset, attaches this information as tags to the data symbols, strips the synchro symbols and passes the stream along. The current implementation employs a least square (LS) estimator with linear interpolation between alternate subcarriers - an MMSE estimator or higher order interpolation will be used for better performance [7], this will require volk kernels for matrix operations.

4.3 Equalization

New kernels as identified in Section 3 will be developed to accelerate the current simple DFE equalizer. A new MMSE-DFE equalizer [8] will be implemented to improve performance in adverse channel conditions.

5 Deliverables

The following is a list of the deliverables which may be subject to change following discussions with mentors.

- Deliverable 0: Provide AVX support for existing VOLK kernels.
- Deliverable 1: Port Turbo and Viterbi encoder/decoder from OpenAirInterfaces to VOLK and upgrade it to AVX.
- Deliverable 2: Vectorize functions like exp, log10, atan2f.
- Deliverable 3: OFDM frame detection kernel, new VOLK kernels based on profiling OFDM, DVB-T, ATSC, Fecapi.
- Deliverable 4: MMSE channel estimator for OFDM.
- Deliverable 5: MMSE-DFE equalizer for OFDM.

6 Schedule

A tentative schedule for the project is as follows :

- **Pre GSoC phase** : Get familiar with AVX. Finalize signal processing blocks to be implemented for OFDM.
- **19 May - 23 May** : Examine the source code, detailed profiling studies.
- **26 May - 30 May** : Upgrade existing VOLK kernels to AVX.
- **2 June - 6 June** : Port OpenAirInterfaces code to VOLK.
- **9 June - 20 June** : Code new volk kernels and test for speedups.
- **23 June - 27 June** : Mid-term evaluations, tie loose ends.
- **30 June - 11 July** : Code OFDM frame detection kernel.
- **14 July - 1 August** : Code new frame synchronizer, channel estimator and equalizer blocks for OFDM (or other signal processing blocks for better performance).
- **4 August - 15 August** : Write tests, bug fixes, clean code, add documentation.

7 Availability

- **Expected hrs/week** : 40 hours per week
- **Off period** : 5-10 blackout days, not contiguous. Lost time will be compensated appropriately.
- **Other commitments** : Possibly one online course (not finalized yet), with 10-15 hrs/week requirement.

8 Background

I finished my B.Tech in Electrical Engineering from IIT Bombay in 2013 and spent the following period working as a research assistant at Carnegie Mellon University, USA. A paper based on my work was accepted for publication at ISCA 2014 (International Symposium on Computer Architecture). I will join a Masters program in Computer Science in August 2014 (school yet to be finalized). My primary research interest is Computer Architecture and I have undergraduate level experience in Signal Processing (courses on Signal/Systems, Communication Systems and Digital Communications). I have significant experience coding in C++ and written some code in Python. You may find some code that I have written previously, on my github : `https://github.com/abhowmick22`. You may find information about my research and my CV at my personal website : `https://sites.google.com/site/abhowmick22`.

8.1 Why GSoC?

My primary motivation is to develop code for a large software project and be part of an open source community - this will help my career objectives.

8.2 Why GNU Radio?

Working with GNU Radio allows me to leverage my electrical engineering background and gives me an opportunity to apply concepts learned in college to real-life scenarios. Also, my interest in developing software for parallel architectures drove me to choose software optimization using SIMD extensions.

9 Conclusion

I would like to acknowledge Nathan West, Martin Braun, Tom Rondeau, Bogdan Diaconescu and the entire community for helping me get started with GNU Radio and advising me on the proposal.

References

[1] "GNU Radio." http://gnuradio.org.

[2] T. Rondeau, "Benchmarking volk in gnu radio." http://www.trondeau.com/blog/2012/2/17/volk-benchmarking.html, 2012.

[3] "Openairinterface." http://www.openairinterface.org/.

[4] "Oprofile, a system-wide profiler for linux systems." http://oprofile.sourceforge.net/about.

[5] T. Schmidl and D. Cox, "Robust frequency and timing synchronization for ofdm," *Communications, IEEE Transactions on*, 1997.

[6] A. Awoseyila, C. Kasparis, and B. Evans, "Robust time-domain timing and frequency synchronization for ofdm systems," *Consumer Electronics, IEEE Transactions on*, 2009.

[7] Y. Shen and E. Martinez, "Channel estimation in ofdm systems - application note." application-notes.digchip.com/314/314-66330.pdf, 2006.

[8] G. Parsace, A. Yarali, and H. Ebrahimzad, "Mmse-dfe equalizer design for ofdm systems with insufficient cyclic prefix," *Vehicular Technology Conference, 2004.*, 2004.