Google Summer of Code 2014 : Performance Optimization with VOLK

Abhishek Bhowmick
abhowmick22@gmail.com

Potential Mentor : Nathan West

March 19, 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, AVX2, FMA, ARM Neon). 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 of the VOLK library with complex kernels and advanced ISA support. First, AVX/AVX2 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 blocks such as the in-tree receiver OFDM implementation, ATSC 2.x, DVB-T. These new kernels will be identified based on profiling studies, to accelerate performance bottlenecks. Also, vectorized support will be developed for transcendental functions such as exp, log10, atan2f etc.

3 SIMD extensions

AVX expands SSE instructions to 256 bit and also includes 3-operand fused multiply and accumulate instructions (FMA). These can be used to speed up operations like dot products, matrix operations, convolution etc. Its extension on Haswell chips, namely AVX2, include newer features like 3-operand general purpose bit manipulation, gather loads and integer vectors. These new features alongwith wider registers can be used with good benefits.

4 Preliminary Profiling

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. AVX2 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* are good candidates for vectorization. Also, profiling was carried out for the *qa_atsc* test in gr-atsc module, which revealed that Viterbi decoder, Trellis encoder, convolutional interleaver/deinterleaver were most CPU-hungry operations.

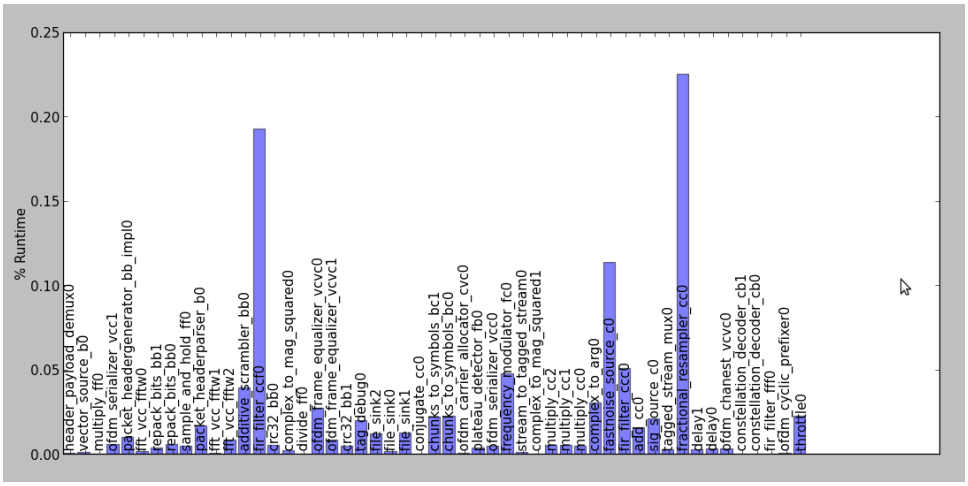

Figure 1: Percentage runtime of different blocks in OFDM RX example

5 New Kernels

Based on the profiling and initial discussions with GNU Radio community, following algorithms were identified for implementation as potential VOLK kernels:

- **OFDM Frame detection**: Current timing and frequency synchronization is done using Schmidl-Cox algorithm [5]. The pre-FFT timing acquisition and fine frequency offset correction will be developed into a single kernel.
- **OFDM Equalizer**: This kernel will incorporate channel estimation based on pilot symbols and DFE equalization.
- **Viterbi Decoder, Trellis Encoder** : These were the most intensive blocks in atsc loopback test. Viterbi decoder was also shown to be the most cycle hungry block in out-of-tree DVB-T implementation by Bogdan Diaconescu [6].
- **Convolutional interleaver/deinterleaver**: Data interleaving and deinterleaving were also seen to be computationally hungry in the profiling of atsc.
- **Transcendental functions**: Operations like log, atan2, exp etc.

6 Deliverables

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

- Deliverable 0: Provide AVX and AVX2 support for existing VOLK kernels.
- Deliverable 1: Port Turbo and Viterbi encoder/decoder from OpenAirInterfaces to VOLK and upgrade it to AVX2.
- Deliverable 2: Vectorize functions like exp, log10, atan2f.

- Deliverable 3: New VOLK kernels like OFDM frame detection, equalizer, Viterbi decoder, Trellis encoder, interleaver/de-interleaver.
- Deliverable 4: Any new kernels as may be suitably identified by profiling.

7 Hardware required

A core Haswell processor from Intel will be required for developing AVX2 and FMA kernels, which I may potentially buy if selected. Also, a USRP board might be needed for testing real-time capabilities of applications like ATSC and DVB-T.

8 Schedule/Availability

A tentative schedule for the project is as follows:

- **Pre GSoC phase**: Get familiar with AVX and AVX2. 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/AVX2.
- **2 June - 6 June**: Port OpenAirInterfaces code to VOLK.
- **9 June - 20 June**: Write kernels for transcedental functions.
- **23 June - 27 June**: Mid-term evaluations, tie loose ends.
- **30 June - 11 July**: Code OFDM frame detection and equalizer kernels.
- **14 July - 1 August**: Code kernels for Viterbi encoder/decoder, convolutional interleaving/deinterleaving and other new operations.
- **4 August - 15 August**: Write tests, bug fixes, clean code, add documentation.

My availability is as follows:

- **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.

9 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`.

9.1 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.

9.2 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.

10 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] "Viterbi decoder implementation for gnu radio dvb-t." http://yo3iiu.ro/blog/?p=1393.