Google Summer of Code 2014 : Performance Optimization with VOLK

Abhishek Bhowmick
abhowmick22@gmail.com

Potential Mentors : Nathan West, Florian Kaltenberger

March 21, 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.

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 OpenAirinterfaces and Existing Kernels

4.1 Porting code from OpenAirInterfaces

OpenAirInterfaces [3] already has SIMD optimized (SSE4) Turbo encoder/decoder and Viterbi encoder/decoder blocks for LTE [4] and IEEE 802.11 [5] standards. The objective is to write generic, aligned and unaligned VOLK wrappers for these blocks, followed by an upgrade to AVX2. The code is licensed under GPLv2 and hence can be ported to GNU Radio. The source code for these blocks may be viewed under `https://svn.eurecom.fr/openair4G/trunk/openair1/PHY/CODING`. The relevant vectorized codes are 3gpplte_sse.c (LTE Turbo encoder), 3gpp_lte_turbo_decoder_sse.c (LTE turbo decoder), viterbi.c (802.11 Viterbi decoder) and viterbi_lte.c (LTE Viterbi decoder).

4.2 Old kernels

Based on the usage of VOLK kernels in various GNU Radio blocks [6] and particularly in applications like digital TV (ATSC, DVB-T), OFDM etc, the following kernels were identified for upgrading to AVX(2).

Will do	Will do given time
volk_32fc_32f_dot_prod_32fc, *volk_32f_x2_dot_prod_32f ...*	*volk_32f_convert_64f,* *volk_8i_s32f_convert_32f ...*
volk_32fc_x2_multiply_32fc, *volk_32f_x2_multiply_32f ...*	*volk_32fc_deinterleave_64f_x2,* *volk_32fc_deinterleave_imag_32f ...*
volk_32fc_conjugate_32fc	*volk_32fc_magnitude_squared_32f*
volk_32fc_x2_multiply_conjugate_32fc	*volk_32fc_s32fc_x2_rotator_32fc*
volk_32fc_s32f_x2_power_spectral_density_32f	

Table 1: Classes of volk kernels identified for updating

5 Preliminary Profiling and New Kernels

Some preliminary 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.

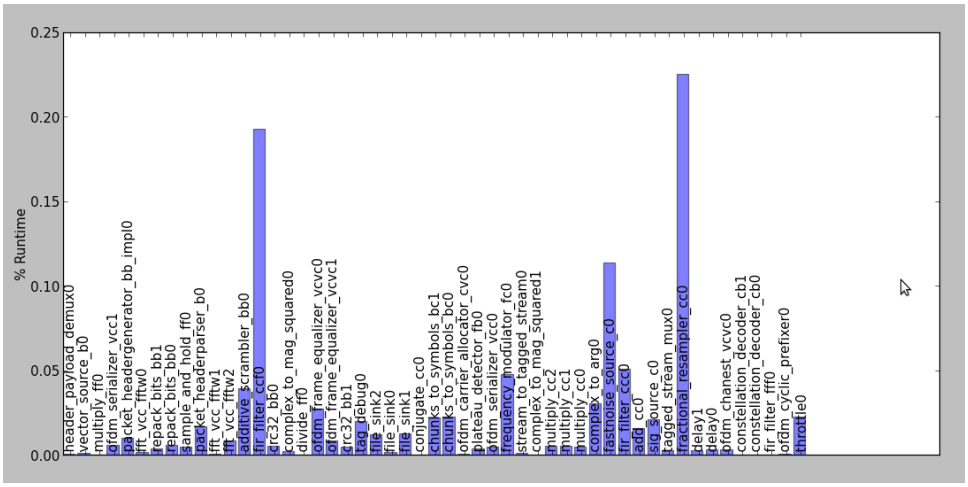

Figure 1: Percentage runtime of different blocks in OFDM RX example

Oprofile [7] 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/encoder, convolutional interleaver/deinterleaver were most CPU-hungry operations.

5.1 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 [8]. 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 among 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 [9]. We can potentially adapt the Viterbi encoder/decoder from OpenAirInterfaces into VOLK kernels for usage in ATSC, DVB-T.
- **Transcendental functions**: Operations like log, atan, exp, sin, cos etc.
- **Convolutional interleaver/deinterleaver**: Data interleaving and deinterleaving were also seen to be computationally hungry in the profiling of ATSC. This will be attempted if time permits.
- **Miscellaneous**: Detailed profiling similar to Section 5 is bound to reveal performance bottlenecks that can be addressed with new kernels. Some operations that were already identified include fractional resampler, fir filter, additive scrambler etc. Identifying and accelerating such operations will be attempted following detailed discussions with mentors and if time permits.

6 Deliverables

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

- Deliverable 1: Upgrade existing VOLK kernels as outlined in Section 4.2 to have AVX/AVX2 proto-kernels.
- Deliverable 2: Port Turbo and Viterbi encoder/decoder for LTE/802.11 from OpenAirInterfaces to VOLK. This will comprise writing generic versions of these blocks, creating volk wrappers and also developing proto-kernels for AVX/AVX2. This will potentially yield Viterbi and Turbo FEC volk kernels that may be re-used by other applications.
- Deliverable 3: Develop new VOLK kernels for OFDM frame detection, OFDM equalizer. In addition to performance, high reconfigurability and portability to other applications such as WLAN, DVB-T, DAB etc will be key concerns.
- Deliverable 4: Vectorize functions like exp, log, atan, sin, cos.
- Deliverable 5: Detailed documentation, QA tests for the above.

- Deliverable 6: (Variable) If time permits, target additional operations such as data interleaving, resampling, filtering, PSK sync. Potential operations to be accelerated will be identified before GSoC begins. Their implementation will be done within GSoC period if time permits, or after program completion as part of continuing association with the community.

7 Hardware required

A Haswell processor from Intel will be required for developing kernels with AVX2 support. 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/AVX2, programming with SIMD intrinsics, detailed profiling to identify performance bottlenecks, engage with the community.
19 May - 23 May (1 week)	Finalize kernels to be implemented, formally define functional specifications
26 May - 6 June (2 weeks)	Upgrade VOLK kernels from Section 4.2 to AVX/AVX2.
9 June - 20 June (2 weeks)	Code OFDM Frame detection kernel.
23 June - 27 June (1 week)	Mid-term evaluations, add documentation, testing, benchmarking.
30 June - 11 July (2 weeks)	Port OpenAirInterfaces channel coding blocks to VOLK and upgrade to AVX/AVX2.
14 July - 18 July (1 week)	Functional tests, benchmarking, documentation.
21 July - 1 Aug (2 weeks)	Code OFDM frame equalizer kernel.
4 Aug - 8 Aug (1 week)	Vectorize exp, log, trig functions.
11 Aug - 15 Aug (1 week)	Write tests, bug fixes, clean code, add documentation.

My availability is as follows:

- **Expected hrs/week**: 40 hours per week
- **Off period**: Approx. 5 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, India in 2013 and spent the following period working as a research assistant at Carnegie Mellon University, USA. A paper I co-authored based on this work was accepted for publication at ISCA 2014 (International Symposium on Computer Architecture). I am currently working as a research staff at IIT Bombay. 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. I also have some experience in x86 assembly and hence would be comfortable with any ASM programming that might be required. 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. I won't receive any university credits for this work - I am participating in order to get experience working on a project with a large codebase, and the whole cycle of software development, testing, maintenance.

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.

9.3 What after GSoC?

Post GSoC, I wish to continue developing more kernels for the VOLK library - finishing off the kernels that were identified but not completed within the timeline of GSoC. I already have some experience with GPU programming, and continuing the work on VOLK will give me significant experience with SIMD programming - such a skill set aligns perfectly with my interest in parallel computing. I am also interested in going beyond vectorization to develop signal processing blocks, however I didn't include it in my proposal keeping in mind the time constraints. As I will be pursuing a Masters degree, I will talk with mentors to work out a realistic plan for continuing my association.

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. I sincerely hope that I get to be a part of the GNU Radio community, either through GSoC or otherwise.

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] "LTE standard for wireless communication." http://www.3gpp.org/technologies/keywords-acronyms/98-lte.

[5] "IEEE 802.11 standard for wireless LANs." http://standards.ieee.org/about/get/802/802.11.html.

[6] "Volk Kernel Benchmarks." http://stats.gnuradio.org/.

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

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

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