GSOC 2014

Alfredo Muniz
muniza@seas.upenn.edu

Using Hardware Based Co-Processors in GNU Radio

Introduction

GNURadio as a digital signal processing program requires numerous mathematical operations to be executed readily and repeatedly. A faster way to process signals in real-time would allow projects with high timing constraints such as channel sounding and spectral estimation. There have been past GNU Radio projects with co-processors including GPUs, FPGAs, and DSPs however their approaches do not scale well with new devices. This summer we would like to implement a purely open source three step approach [1] that would improve the state of GNU Radio co-processing for years to come.

GNU Radio and Co-Processors

At GRCon 2011, there was gr-gpu by the University of Maryland [2]. At GRCon 2012, there was GReasy by Virginia Tech [3]. At GRCon 2013, there was gr-fpga as a GSoC project, gr-dsp by Sandia National Labs, a talk by Ettus Research on the current state and future of GNU Radio co-processing, and a working group coproc was formed to continue the conversation on how to best use GNU Radio with co-processors [4]. Unfortunately after the conference, there was only one coproc call [5] and afterwards very few conversations on the topic [6]. The problem is clearly not one of interest but of accessibility.

So far the methods developed for co-processor work have required using closed source software. For instance TI's Code Composer Studio (CCS) to program DSPs, Xilinx's Integrated Software Environment (ISE) to program FPGAs, or NVIDIA's Compute Unified Device Architecture (CUDA) to program GPUs. A completely open source approach is required for accessibility and a long term solution to GNU Radio co-processing.

The gr-gpu project required rewriting the processing portion of the blocks to CUDA computer kernels [7]. This does not scale well as every block that needs to be accelerated would require a rewriting in the CUDA platform. The gr-fpga project is the same as they require each block to be written in verilog or VHDL for it to be accelerated [8].

Some ideas in gr-dsp [9] will be used in this project but the implementation is different. First, we are using an DSP+ARM SoC, specifically the TCI6638K2K [10], whereas Sandia used a DSP connected to a host pc through PCIe. Our implementation has shared memory so we do not have to rely on data copying as they did. They used Ti's software whereas we will be going for a completely open source approach. This includes writing our own ARM toolchain and DSP runtime libraries.

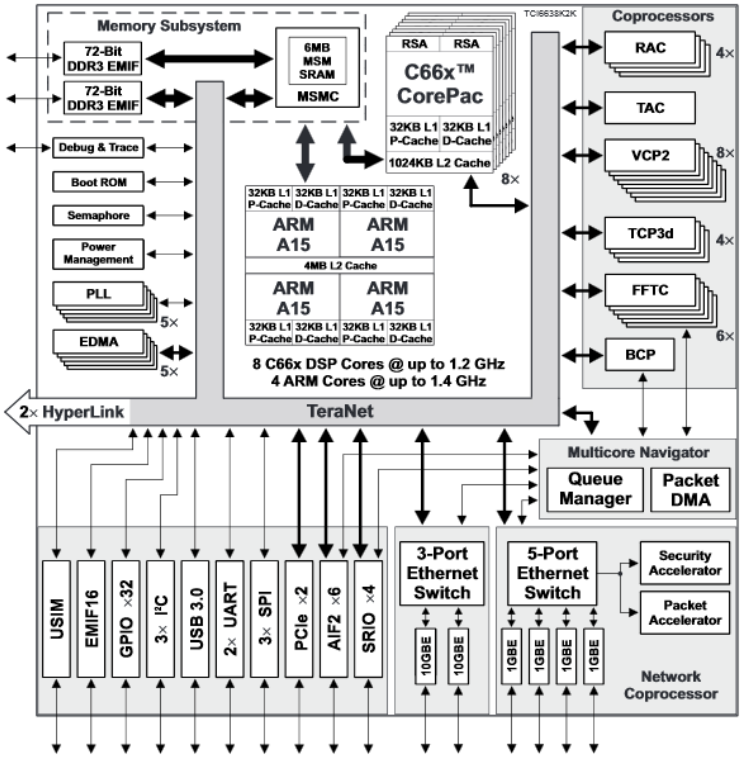

Figure 1: Functional Diagram of TCI6638K2K

The TCI6638K2K on the Keystone2 uses the A15 which is 40% more powerful than the A9 used in gr-fpga, and it uses double the amount of cores and coprocessors than the TMS320C6670 used in gr-dsp. In addition there is shared memory so we don't need to perform data copies between devices. Our specific model in addition to the DSP has multiple dedicated co-processors including Fast Fourier Transforms, Viterbi Decoders, and Turbo Decoders as seen in figure 1. The availability of multiple co-processors on a single chip make it the perfect candidate for testing the scheduler in the three step approach.

For GSoC, we would like to focus on the three step approach mentioned earlier. The first is to provide support for GNU Radio to run blocks on a co-processor. This requires the user to select parts of the flow graph and assign them to a particular co-processor. This will likely take the form of Egress and Ingress used in gr-dsp. GNU Radio would then partition the flow graph into sub-flow graphs while inserting TCP sources and sinks for the interconnects between sub-flow graphs. Each sub-flow graph is self contained and would generate its own python script that manages the communication between the host and the coprocessor.

The second step is to extend support to the co-processors which involves adding the liquid-dsp library to the DSP, and establishing stable communication between the ARM and the DSP. Although we require rewriting blocks, most of the processing is done for us in the liquid-dsp library so integration would require a wrapper which is more manageable than writing each block in a different language. A candidate for the communication is to use pthread to create a DSP thread on an available core and to also send back an interrupt once the program is complete. This involves compiling DSP code on the fly and using ping-pong double buffer to swap buffers once interrupts are triggered to maximize throughput. This is currently being worked on at Virginia Tech [11].

The third step is to develop a unified scheduler capable of dynamically partitioning processing tasks and assigning them to suitable processors. The method we will explore would require writing a control loop that dynamically adjusts itself based on the time it takes to process through the co-processors. This ignores lots of parameters such as non-local memory (NUMA) issues and processor setup overhead. Because of the multiple co-processors present on the board, we are able to test this on a small scale. The goal is that once we have validated the three step approach, adding new co-processors including FPGAs and GPUs to GNU Radio in the future will be easier.

Deliverables and Timeline for 13 Weeks

Preliminary Work (03/08/14 – 05/19/14): Communicate with mentor about things I need to know before starting the summer. This includes getting up to speed on the new buffer technique that allows us to share data without copying samples between architectures and getting familiar with the keystone2 work done so far at Virginia Tech.

Setting up the Platform for 2 weeks (05/19/14 – 06/02/14): Make GNU Radio Co-processing more accessible by providing a purely open source Installation. This would require working together with Virginia Tech on the gr-dsp project currently on github and testing it on the keystone2. Success depends on a working GNU Radio installation on the A15s and dsp functions running on the DSP.

GNU Radio Integration Part I for 3 weeks (06/02/14 – 06/23/14): Begin GNU Radio Co-processing framework. This includes adding the ingress and egress blocks as well as a couple of keystone2 dsp blocks that would be used to test the implementation. Success depends on being able to offload tasks to the DSP and accelerate functions gr-dsp_fxn.

Midterm Evaluations for 1 week (06/23/14 – 06/30/14): Turn in the Google midterm evaluation by the 27th. Consider everything that has been done and evaluate with mentor if plans need to be changed. Do thorough documentation on what has been done so far.

GNU Radio Integration Part II for 3 weeks (06/30/14 – 07/21/14): Continue GNU Radio framework. This includes implementing the shared memory buffer technique (at this time it should be working on the zynq) onto the Keystone2. Integrate the other co-processors available on the keystone2 with the buffer technique. Success depends on co-processor specific blocks – gr-dsp_fftc_fxn.

Preliminary Work on Scheduler for 3 weeks (07/21/14 – 08/11/14): Combine the DSP and co-processor blocks developed before to make a new block that automatically runs samples through each co-processor. Start simple by using the latency to automatically switch which co-processor to use for the specific task. Success depends on a working new block/function – gr-dsp_optimize_fxn.

Final Evaluations and Code Samples for 1 week (08/11/14 – 08/22/14): Turn in the Google final evaluation and code samples by the 22nd. Write extensive documentation including a tutorial on the installation and examples one can use the co-processor for on the GNU Radio wiki so others can follow the process. Get started on the presentation for the GNU Radio conference.

Communication

I have already established communication with Philip Balister and we have added each other on Google Plus to have hangouts. I plan to communicate with him any questions I have through email as they arise and update him weekly on the status of the project through hangouts. I want to send a weekly email to the list to keep everyone updated on the project and to receive feedback. I also want to take part in all the GNU Radio developer calls to learn more about the other projects going on and to see where I best fit in after the program.

About Me

My name is Alfredo Muniz. I am a third year undergraduate student from Houston, Texas studying Electrical and Mechanical Engineering at the University of Pennsylvania. I run the amateur radio club at my university and run the only VE team in Philadelphia. I teach students skills such as how to make PCBs and how to use GNU Radio. I read technical books on my free time such as Hacking by Erickson and Digital Signal Processing by Smith. This project has no connection with university research and bears no credit towards my degree. I can get the evaluation board from my school (no strings attached) but do not have direct access to USRPs. Ben from Ettus Research has told me there is a possibility of collaborating at their headquarters during the summer.

I first began my journey into Software Defined Radio last summer when working on a research project at my university for improving the state of electric vehicles. There were about six people in the lab all working on entirely separate projects from low power wireless protocols to pacemaker verification. There was no community and no collaboration. In short, last summer was a miserable experience. That's when I realized that I needed to find what I want instead of having someone place me on projects based on words on a paper. I took a week off and began with hacking. I watched over 20 hours of presentations from DEFCON and was especially intrigued by RenderMan's Hackers + Airplanes. That's when I bought an RTLSDR dongle and built an antenna for ADS-B. That's when I first used GNU Radio. For days I was watching videos, going through examples, and trying all sorts of out of tree modules to explore the magic of GNU Radio.

A few months later I applied to the GNU Radio Conference for the student travel award and surprisingly with my little bit of experience was given the opportunity! I had a really amazing time and couldn't believe that a community of people who enjoy learning and experimenting actually existed. That was when I decided I wanted to become a part of this community and so have joined the GNU Radio Tutorials Task Force [12] to write tutorials for beginners.

Coming from an electrical engineering background, I was most motivated by the co-processing presentations during the conference. I want to dedicate the summer to improving the state of GNU Radio co-processing and to gain enough experience to be able to contribute to the work of others. After the summer, I want to bring back coproc and take an active role in accelerating GNU Radio by participating in the coproc dev calls and reporting back to the general GNU Radio dev calls.

My previous coding experience can be found on my github [13]. Most of my code is in C and microcontroller based so no large scale projects where revision control was needed. I have worked on multiple team projects involving the synthesis of mechanical, electrical, and computer science skills [14]. I have gone through examples in GNU Radio and am comfortable writing new blocks in python and C++. I have been on the mailing list for a little over half a year now and have answered a couple of questions. My biggest contribution so far is helping write the new tutorials.

All code produced will be GPLv3 licensed and made publicly available on github. I will work at least 40 hours a week and any breaks will be on the weekends. This project will be my main priority during the summer. This is the only proposal I am submitting to GSoC and I will be interning at Ettus Research if my proposal is not accepted.

References

[1] Link to paper on co-processors written at Virginia Tech. Most of this proposal is based off the paper.
https://github.com/GRDSP/grdsp/blob/master/docs/grsoc.pdf

[2] GNU Radio 2011 Conference Abstracts and Presentations
http://gnuradio.squarespace.com/grc2011-abstracts/

[3] GNU Radio 2012 Conference Abstracts and Presentations
http://gnuradio.squarespace.com/grc2012-abstracts/

[4] GNU Radio 2013 Conference Abstracts and Presentations
http://gnuradio.squarespace.com/grcon2013-presentations/

[5] Summary of co-processor working group call
http://gnuradio.org/redmine/projects/gnuradio/wiki/GRCon13Coprocessor

[6] Mailing List Archives on the state of the co-processor working group
http://lists.nongnu.org/archive/html/discuss-gnuradio/2014-02/msg00171.html

[7] Applying Graphics Processor Acceleration in a Software Defined Radio Prototyping Environment
http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5929977&tag=1

[8] FPGA Accelerators in GNU Radio with Xilinx's Zynq System on Chip
http://gnuradio.org/redmine/projects/gnuradio/wiki/Zynq

[9] Using Digital Signal Processors in GNU Radio
http://gnuradio.squarespace.com/storage/grcon13_presentations/grcon13_ford_snl_dsps.pdf

[10] The DSP+ARM Keystone2 SoC Used in the Advantech EVMK2HX
http://www.ti.com/product/tci6638k2k

[11] Github to the Work Currently Being done at Viginia Tech
https://github.com/GRDSP/grdsp

[12] Tutorials Task Force. See Tutorial 2 & 3.
http://gnuradio.org/redmine/projects/gnuradio/wiki/TutorialsTaskForce

[13] Github to my past experience in programming
https://github.com/muniza

[14] Website to Projects I have worked on with a friend outside of school work
http://www.sadeoba.com/#!projects/c1p9k