Title: Raspberry Pi 2B+ runs FFT on GPU and CPU, energy and power comparison

Author: Qihao He, Bruce Segee, Vincent weaver

# GPU part:

Library name: GPU\_FFT

Written by: Andrew Holme

## Library description:

Reference:

ACCELERATING FOURIER TRANSFORMS USING THE GPU

<https://www.raspberrypi.org/blog/accelerating-fourier-transforms-using-the-gpu/>

General purpose code for the VideoCore IV graphics processing unit(GPU) in the BCM2835. To create an accelerated Fast Fourier Transform library. Taking the Fourier transform of a function yields its frequency spectrum (i.e. the pure harmonic functions which can be added together to reconstruct the original function).

GPU\_FFT is an FFT library for the Raspberry Pi which exploits the BCM2835 SoC V3D hardware to deliver ten times the performance that is possible on the 700 MHz ARM. Kernels are provided for all power-of-2 FFT lengths from 256 to 131,072 points inclusive.

GPU\_FFT uses single-precision floating point for data and twiddle factors, so it does not compete on accuracy with double-precision libraries; however, the relative root-mean-square (rms) error for a 2048-point transform is less than one part per million, which is not bad.

The library runs on dedicated 3D hardware in the BCM2835 SoC, and communication between ARM and GPU adds 100µs of latency which is much longer than the shortest transform takes to compute! To overcome this, batches of transforms can be executed with a single call. Typical per-transform runtimes in microseconds are:

## ­Library usage:

Being able to perform lots of Fourier transforms quickly is useful for all sorts of audio and radio applications including, unsurprisingly, GPS. Ham radio enthusiasts will also find Andrew’s work very useful.

Last October, Eben attended the Radio Society of Great Britain (RSGB) Convention, where radio amateurs told him they wanted a speedy fast Fourier transform (FFT) library to do Software Defined Radio (SDR) projects on the Pi.

## Library operate:

To get GPU\_FFT enter the following at the command prompt:

sudo rpi-update && sudo reboot

To build and run the example program:

cd /opt/vc/src/hello\_pi/hello\_fft

make

sudo mknod char\_dev c 100 0

sudo ./hello\_fft.bin

API documentation can be found in the hello\_fft folder.

File: hello\_fft usage:

$ sudo ./hello\_fft.bin

Usage: hello\_fft.bin log2\_N [jobs [loops]]

log2\_N = log2(FFT\_length), log2\_N = 8...22

jobs = transforms per batch, jobs>0, default 1

loops = number of test repeats, loops>0, default 1

e.g.

$ sudo ./hello\_fft.bin 8

rel\_rms\_err = 3.3e-07, usecs = 45, k = 0

$ sudo ./hello\_fft.bin 22

rel\_rms\_err = 1.5e-06, usecs = 782379, k = 0

$ sudo ./hello\_fft.bin 8 2 2

rel\_rms\_err = 3.1e-07, usecs = 33, k = 0

rel\_rms\_err = 3.1e-07, usecs = 23, k = 1

## Library results:

# CPU part:

Numpy