Gmail by Google

Graham Jones <grahamjones139@gmail.com>

# Re: [SYLT-FFT] Compilation for other Processors (#1)

**D Taylor** <notifications@github.com> 13 March 2015 at 08:56
Reply-To: stg/SYLT-FFT
<reply+0004b963696013ebecf9aba2eedb00dd435c73bb5fde134392cf00000001111a685992a169ce0342ca16@reply.github.com>
To: stg/SYLT-FFT <SYLT-FFT@noreply.github.com>
Cc: Graham Jones <grahamjones139@gmail.com>

Thank you for writing me, this is exactly the sort of thing that makes one happy after releasing free code.
Your project seems very useful and important and even though my contribution is tiny, it gives me great satisfaction to have my code be part of it.
I am also impressed by the broad spectrum of solutions, languages and platforms you have tried.

Read through your source today (which seems very nicely written by the way), and while I am sure performance or memory is not much of a concern in this case you can speed up the FFT calculation significantly while also reducing memory requirements:

```
for(i = 0; i < NSAMP / 2; i++) {
fftdata[i].r = accData[i * 2 + 0];
fftdata[i].i = accData[i * 2 + 1];
}
fft_fftr(fftdata, FFT_BITS - 1);
```

You are working with REAL-only data (.i is always 0) and this allows for optimizing an FFT calculation in such a way that you are able to process your REAL data using an FFT of half the size in the COMPLEX domain, reducing computation time to less than half.
This is offset by the need to perform some conversions after the FFT, but there is still a good net gain.

If you have no need for accData after converting, you can actually do the real FFT version directly on accData given that it's data format is changed to int32:

```
fft_fftr((fft_complex_t *)accData, FFT_BITS - 1);
```

After which you could still do:

```
maxVal = getMagnitude(((fft_complex_t *)accData)[1]);
```

This would save you even more memory and processing time.

This is all probably of little interest to you, but I figured you should have the information, even if for future reference only.

What may be of greater interest to you though, is windowing of the FFT.
I believe your output will be more useful if you implement a suitable windowing method.
Currently you are technically using a rectangular window which is rarely optimal for any kind of continuous signal.
If you are interested, National Instruments has a nice introductory document at http://www.ni.com/white-paper/4844/en/

Best regards,
Davey

[Quoted text hidden]

> > <https://github.com/stg/SYLT-FFT/issues/1#issuecomment-71525495>.
> >
>
>
>
> --
> Graham Jones
> Hartlepool, UK.

> 
> —
> Reply to this email directly or view it on GitHub
> <https://github.com/stg/SYLT-FFT/issues/1#issuecomment-78368873>.
> 

--

[Quoted text hidden]