Efficient Real-Time Filter Design for Recording Multichannel Neural Activity

Josh Chartier (jmc12@rice.edu), Hasitha Dharmasiri (hkd1), Chelsea Rodrigues (cmr8), Vivas Kumar (vk7)

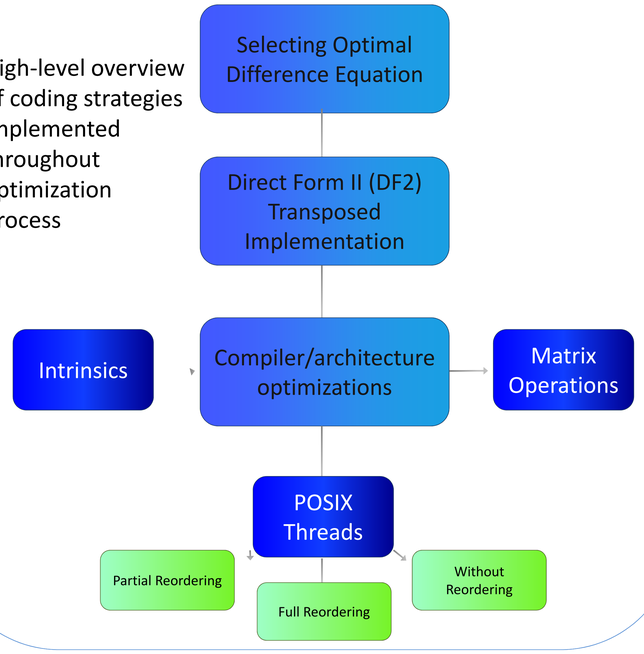
Motivation

We investigated the optimization of multi-channel filtering signals to overcome current limitations in real-time signal processing of neural recordings.

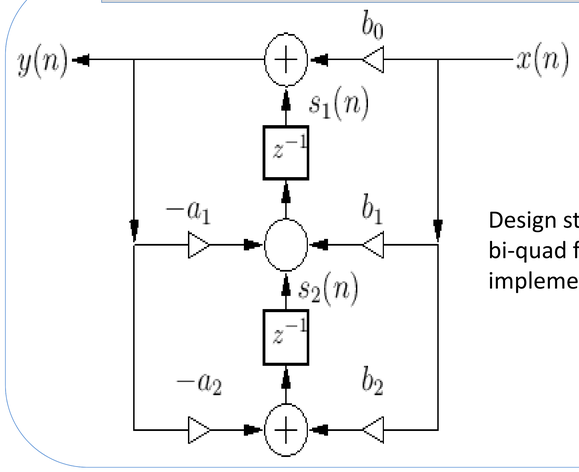
Open Ephys GUI, an open source platform for analyzing recorded neural signals in real time with which we work, requires heavy CPU usage when processing >32 signals simultaneously.

Methodology

High-level overview of coding strategies implemented throughout optimization process



Direct Form II Transposed Structure



Design structure for bi-quad filter for code implementation

Test Parameters

All data was collected on a laptop equipped with an AMD A6-3400M quad-core processor, which supports a clock rate of up to 2.3 GHz, under these conditions:

- Implemented filter: 4-pole Butterworth bandpass filter
- Number of input channels = 256
- Number of time samples = 600,000
- Number of data filter cycles = 100

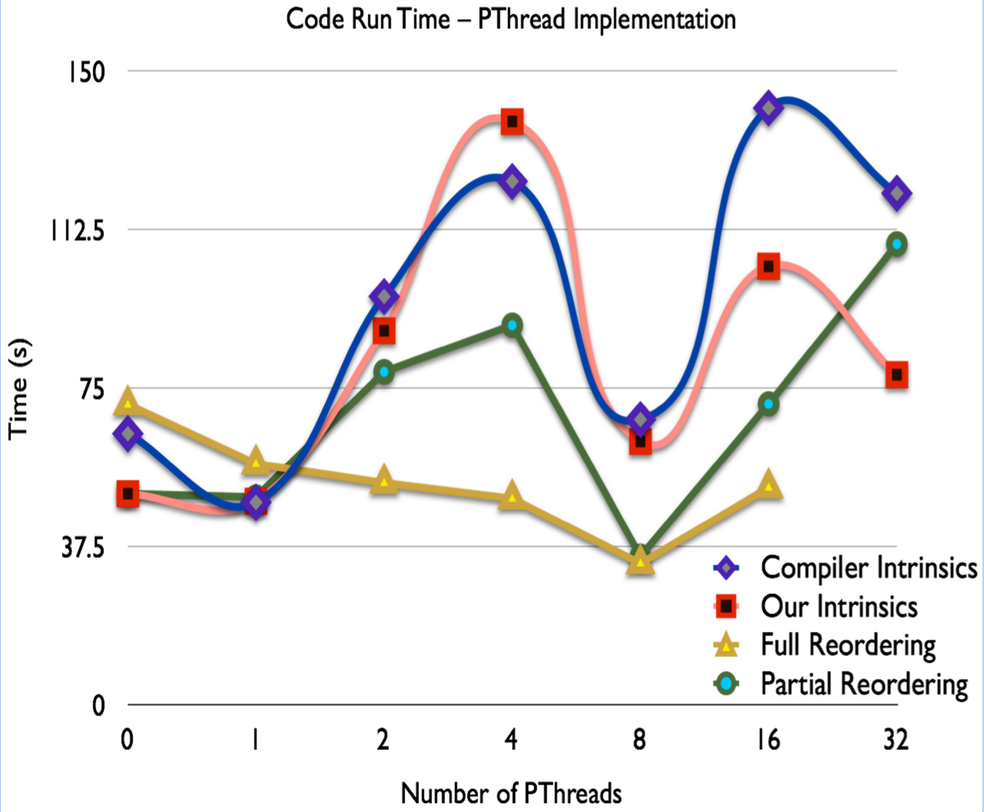


Figure 1. Total Code Operation Time – PThread Implementation. 4 tests under various PThread control conditions demonstrate similar patterns according to the number of PThreads used during code execution.

- Compiler Intrinsics – Implementation with naïve data structures relying purely on compiler optimizations of SSE3 intrinsics
- Our Intrinsics – Implementation with naïve data structures using our custom SSE3 intrinsics code
- Partial Reordering – Implementation assigning an output vector to each thread utilizing compiler intrinsics
- Full Reordering – Intermediate variables are aligned by channel and separated by thread

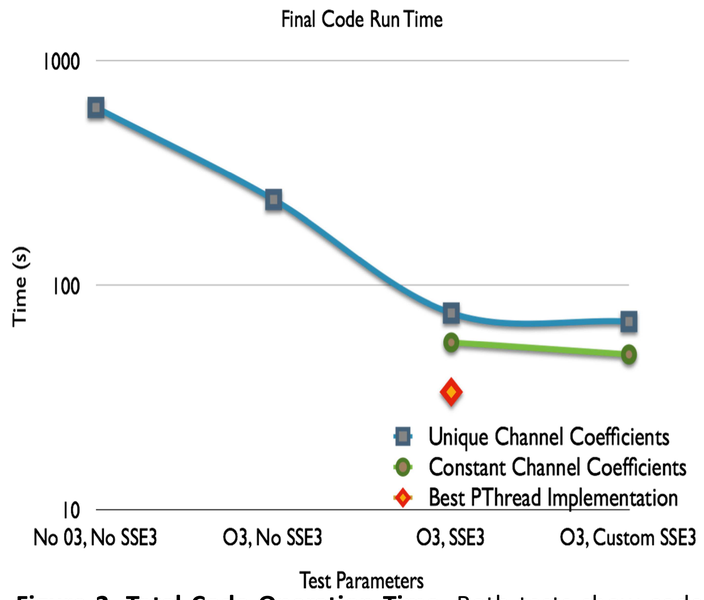


Figure 2. Total Code Operation Time. Both tests show code execution times based on different methods of compiler optimization. Default features included slow arrangement of data and 4 separate w vectors for temporary variables. Test 1 had a & b coefficients in vectors; Test 2 fixed coefficients for all channels.

Conclusions and Future Work

Our optimal filter implementation utilized the DF2t filter structure with 8 PThreads and innovative data structuring with SSE3 intrinsics instruction set. Results suggest that 8 PThreads provide the perfect combination between cache hits, CPU utilization, and no cache poisoning. Based on our code implementation results, there are three key areas for future directions for this project.

- Optimizing code for GPGPU OpenCL framework
- Analysis of multiple architectures for optimized filter performance
- Integrate code into open source Open Ephys GUI

Acknowledgements

We thank Dr. Richard Baraniuk, Eva Dyer, Dr. Caleb Kemere, and Professor Ray Simar for their advice and contribution to our project. We also thank the Department of Electrical and Computer Engineering at the Brown School of Engineering, Rice University.