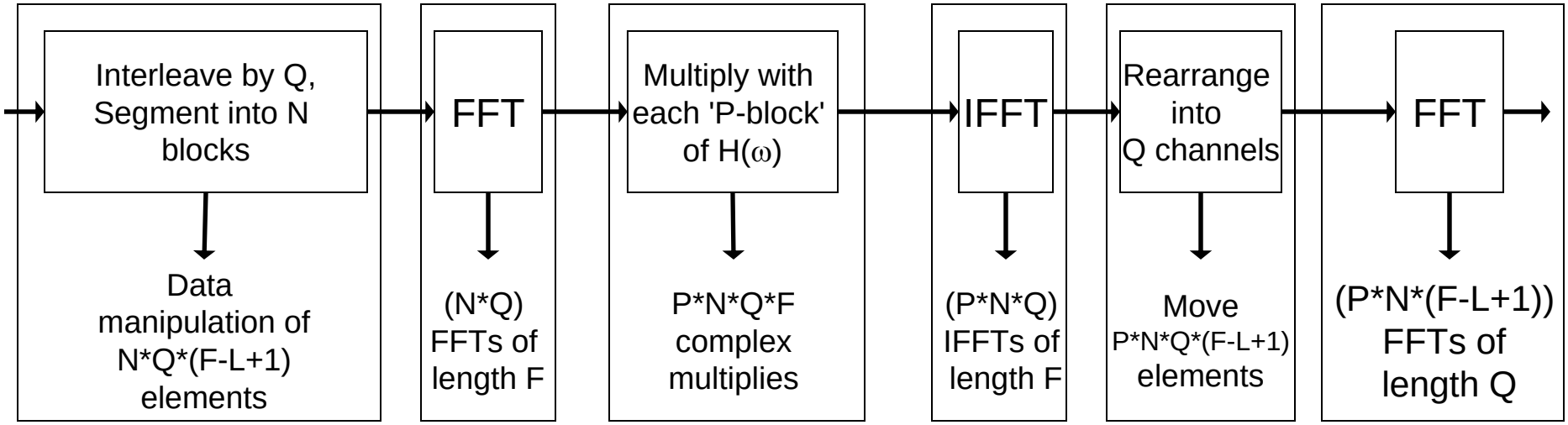
Channelization and Resampling Using a Graphics Processing Unit

SAIC
September 23rd, 2008

Overview

- Channelization and resampling of digital signals are common elements in Software-Defined/Cognitive Radio Applications.
- However, channelization and resampling consume a significant portion of the overall system computation budget
- This can limit the data rates at which the system can operate.
- SAIC's polyphase GPU implementation was typically on the order of 4-10 times faster than the associated CPU implementation.

Polyphase GPU Implementation



Example:
P=1,Q=64,
L=31,N=73,
F=256
(~0.0276 sec
on 8800GTX
incl file I/O)

P: Upsample Rate
Q: Downsample Rate
L: (Filter Length / (P * Q))
N: Number of input blocks to process in parallel
F: FFT length

- FFT length determined by filter length
  - This is related to max(P, Q) / min(P, Q)
- Number of parallel input blocks N is constrained by memory and computing resources
- ~83% of time spent in data conversion and file I/O for this example

Implementation Details

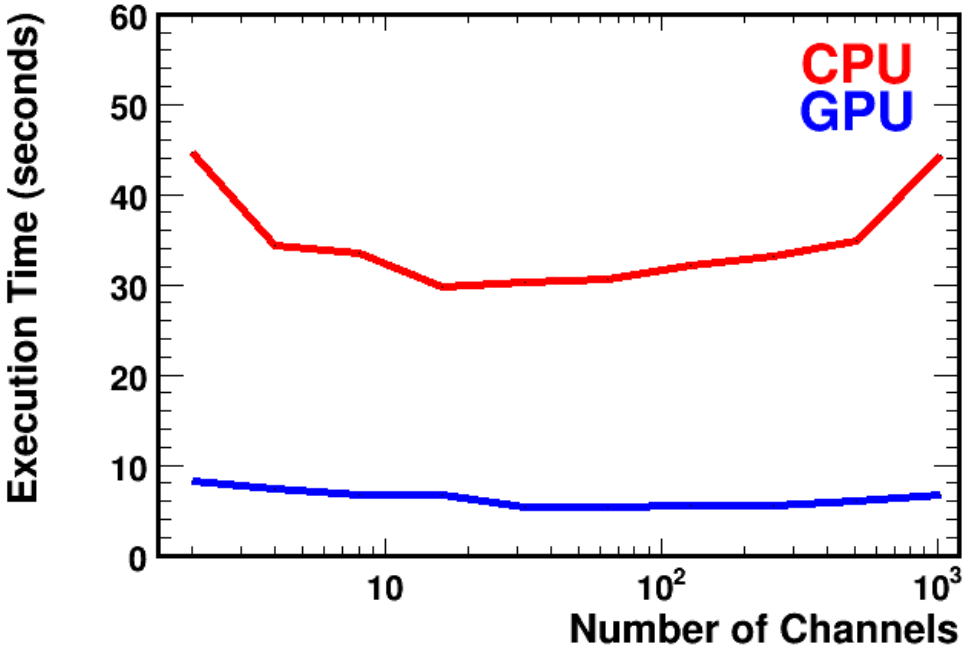
Host (CPU) implementation:
•Uses fftw floating-point libraries (3.1.2, not multi-threaded)
•Uses parallel dft plans

GPU implementation:
•Uses NVIDIA's CUFFT library for parallel FFTs
•Custom kernels for data manipulations, block multiplies, and floating-point conversion
•Tests run on 8800 GTX card under RHEL5 (32-bit)
•8800 GTX has 16 multiprocessors
•Maximum active threads = (16*768) = 12,288, each running at 1.35Mhz

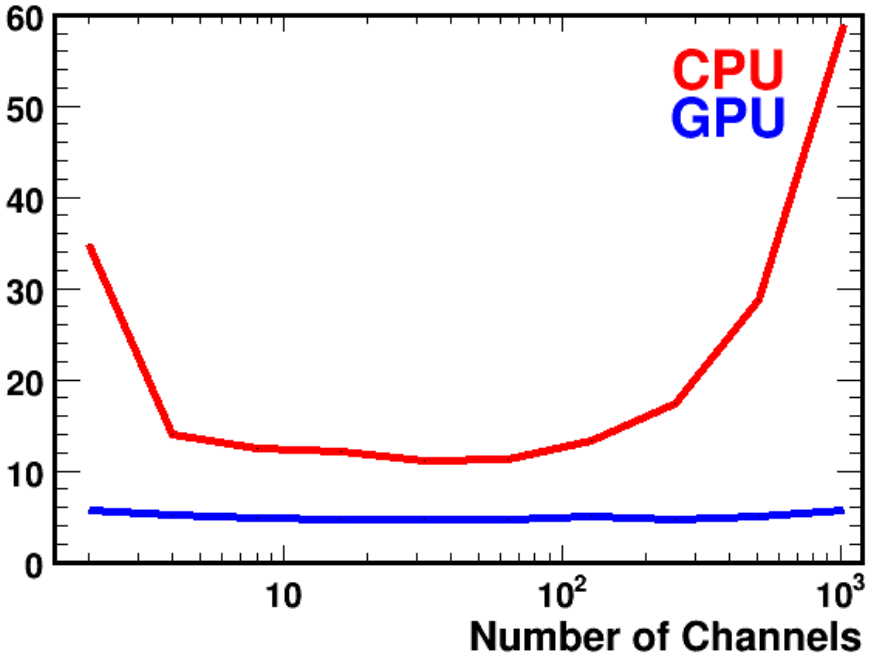
General Notes:
•All timing results include reading data from the file
•GPU timing results include transferring data to and from the GPU
•All computations use single-precision IEEE-754 arithmetic

Results

GPU implementation was typically on the order of 4-10 times faster than the associated CPU polyphase implementation across a wide range of channelization levels.



Channelization Time for 1GB 16-bit Complex Signed Data File



Resample Time for 1GB 16-bit Complex, Signed Data File