GPU VSIPL: Core and Beyond

Andrew Kerr[1], Dan Campbell[2], and Mark Richards[1]

[1]Georgia Institute of Technology
[2]Georgia Tech Research Institute

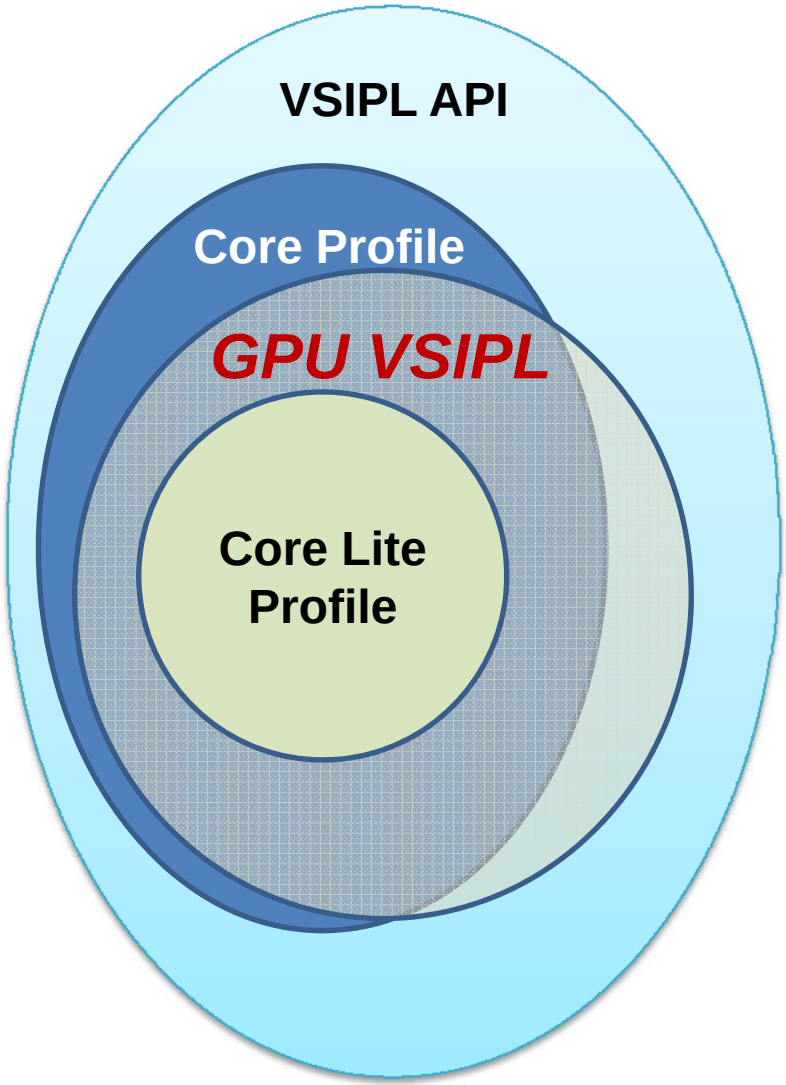
Goal

- **An application development environment for embedded high performance computing that achieves**
  - *Portability*: ***same code usable with different processors, processor generations, and vendors***
  - *Productivity*: ***disciplined programming model, leverage highly optimized libraries for signal processing+linear algebra***
  - *Performance*: ***employ highly advanced processors, ~1 TFLOPS***

Approach

- **Adopt the VSIPL API for open standard portability and productivity**
- **Develop a state-of-the-art GPU-VSIPL library to leverage CUDA-enabled GPU performance**

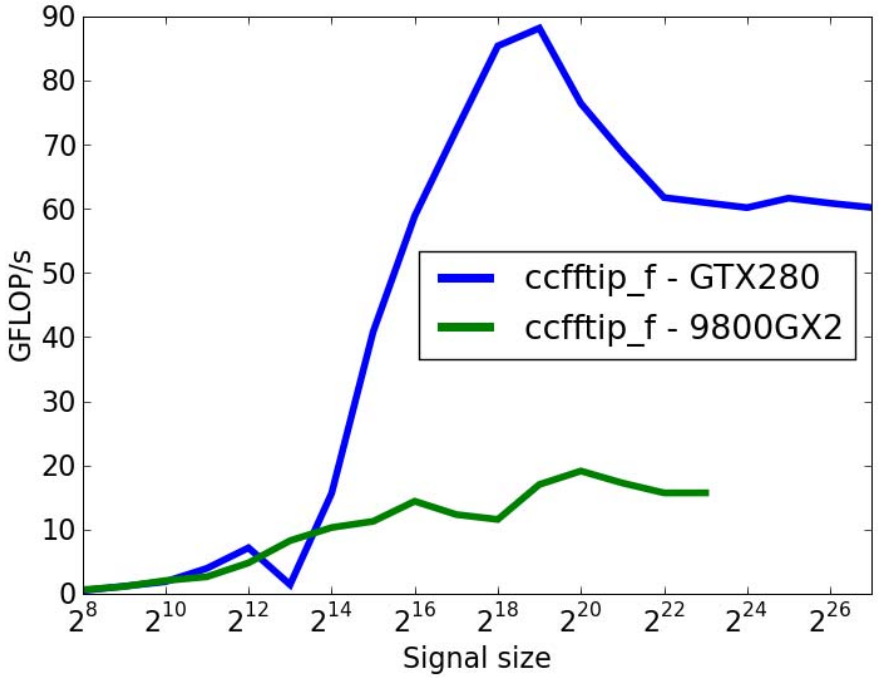
GPU-VSIPL Functional Coverage



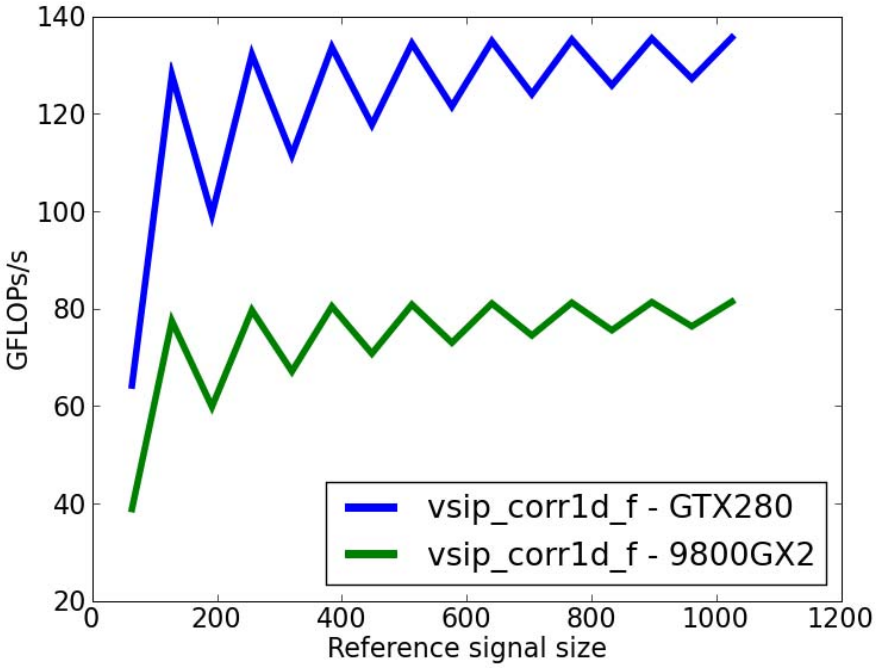
- **What's covered from VSIPL Core**
  - *Data Types*
    - real, complex, integer, boolean, index
  - *View Types*
    - Matrix, vector
  - *Element-wise Operators*
    - arithmetic, trigonometric, transcendental, scatter/gather, logical, and comparison
  - *Signal Processing*
    - FFT (in-place, out-of-place, batched)
    - Fast FIR filter, window creation, 1D correlation
    - Random number generation, histogram
  - *Linear Algebra*
    - generalized matrix product
    - QR decomposition, least-squares solver
- **What's Not (yet)**
  - *Linear Algebra*
    - LU, Toeplitz, least-squares solvers
- **What's Added Beyond VSIPL Core**
  - *Scalar and matrix versions of element-wise vector operators*
  - *Matrix utility functions*

Performance Examples: Signal Processing

1D FFT, In-Place

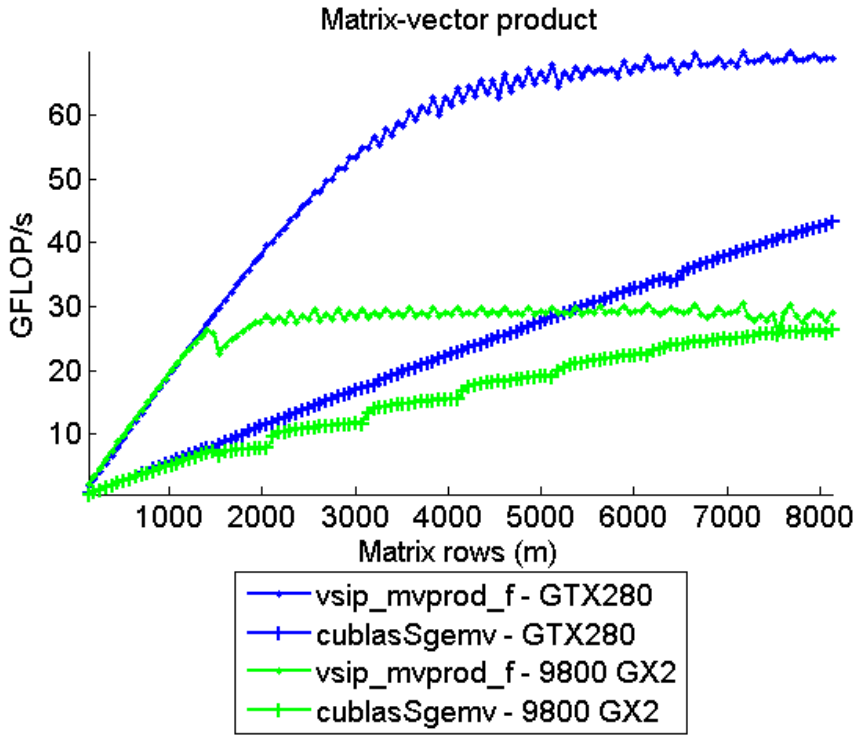


1D Correlation



Performance Examples: Linear Algebra

Matrix-Vector Product



QR Decomposition