

# GPU VSIPL: High Performance VSIPL Implementation for GPUs

**Andrew Kerr, Dan Campbell\*,  
Mark Richards, Mike Davis**

[andrew.kerr@gtri.gatech.edu](mailto:andrew.kerr@gtri.gatech.edu), [dan.campbell@gtri.gatech.edu](mailto:dan.campbell@gtri.gatech.edu),  
[mark.richards@ece.gatech.edu](mailto:mark.richards@ece.gatech.edu), [mike.davis@gtri.gatech.edu](mailto:mike.davis@gtri.gatech.edu)

## High Performance Embedded Computing (HPEC) Workshop

24 September 2008

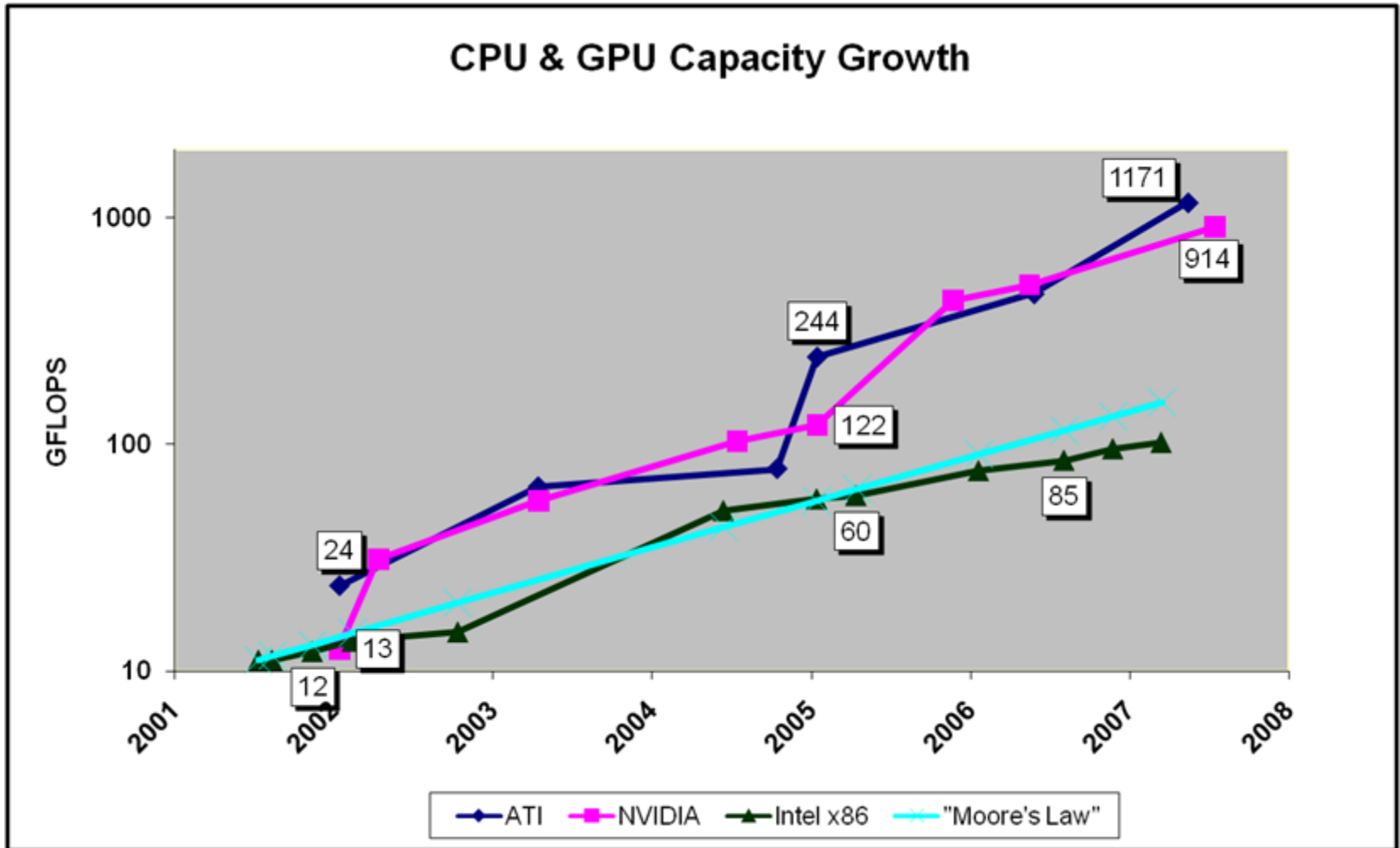
Distribution Statement (A): Approved for  
public release; distribution is unlimited

This work was supported in part by DARPA and AFRL  
under contracts FA8750-06-1-0012 and FA8650-07-C-  
7724. The opinions expressed are those of the authors.

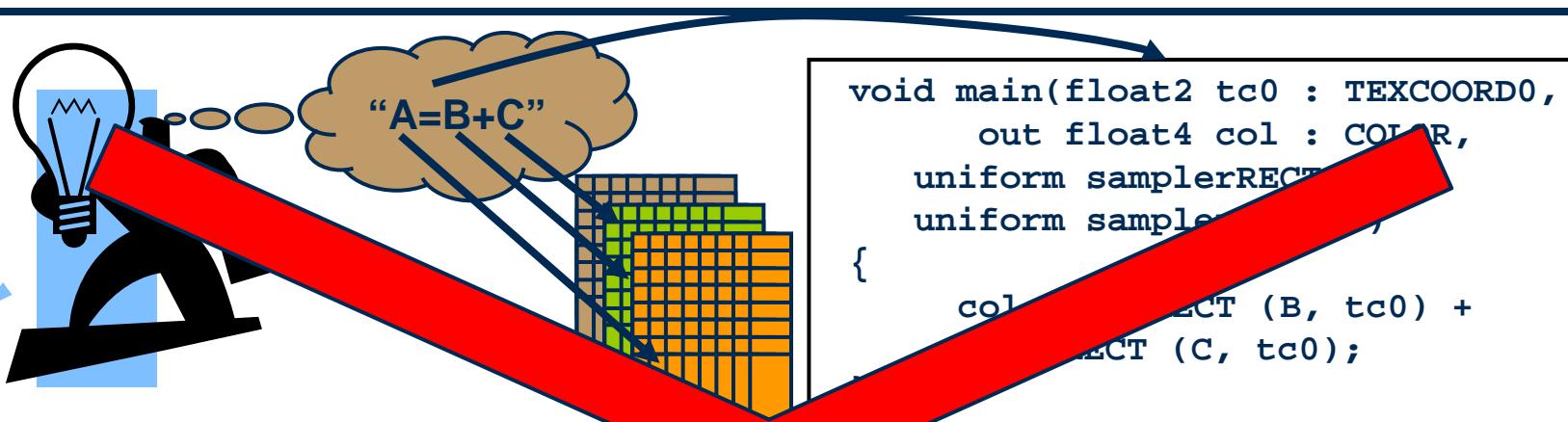
# Signal Processing on Graphics Processors

- GPUs original role: turn 3-D polygons into 2-D pixels...
- ...Which also makes them cheap & plentiful source of FLOPs
  - Leverages volume & competition in entertainment industry
  - Primary role highly parallel, very regular
  - Typically <\$500 drop-in addition to standard PC
- Outstripping CPU capacity, and growing more quickly
  - Peak theoretical ~1TFlop
  - Power draw: 280GTX = 200W Q6600 = 100W
  - Still making improvements in market app with more parallelism, so growth continues

# GPU/CPU Performance Growth



# GPGPU (Old) Concept of Operations



- Arrays  $\rightarrow$  Textures
- Render polygons in the same pixel dimension as output texture
- Execute fragment program to perform desired calculation
- Move data from output buffer to desired texture

Now we have compute-centric programming models...  
... But they require expertise to fully exploit

# VSIPPL - Vector Signal Image Processing Library

- **Portable API for linear algebra, image & signal processing**
- **Originally sponsored by DARPA in mid '90s**
- **Targeted embedded processors – portability primary aim**
- **Open standard, Forum-based**
- **Initial API approved April 2000**
  
- **Functional coverage**
  - **Vector, Matrix, Tensor**
  - **Basic math operations, linear algebra, solvers, FFT, FIR/IIR, bookkeeping, etc**

# VSIPPL & GPU: Well Matched

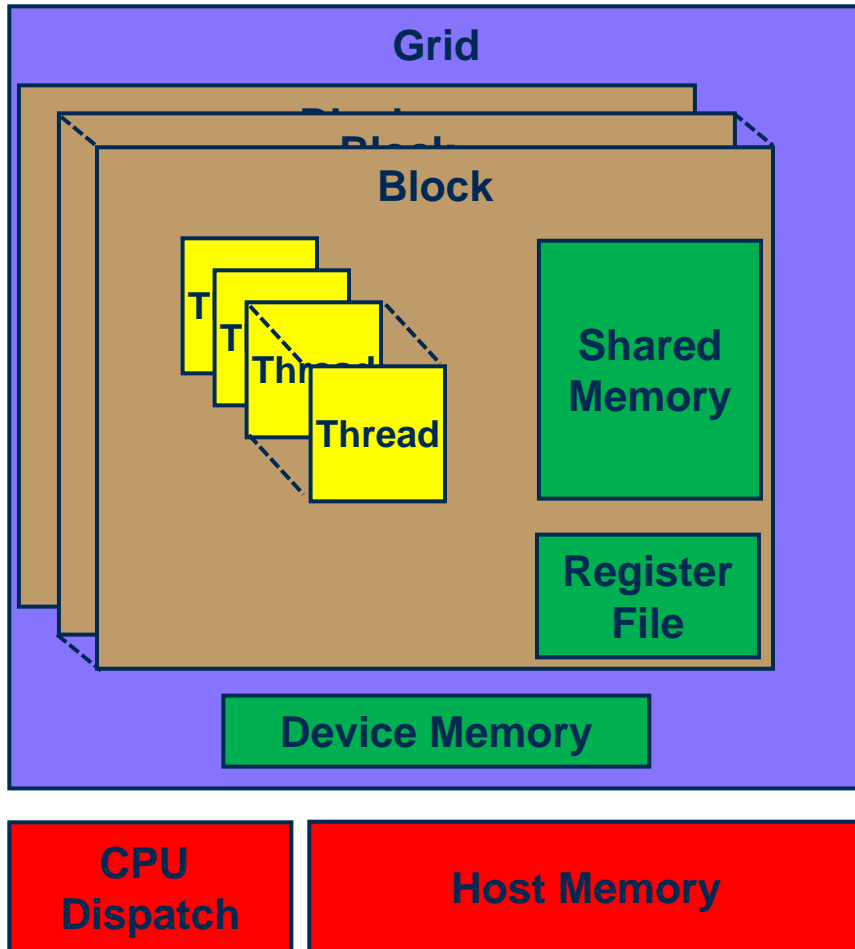
- **VSIPPL is great for exploiting GPUs**
  - High level API with good coverage for dense linear algebra
  - Allows non experts to benefit from hero programmers
  - Explicit memory access controls
  - API precision flexibility
- **GPUs are great for VSIPPL**
  - Improves prototyping by speeding algorithm testing
  - Cheap addition allows more engineers access to HPC
  - Large speedups without needing explicit parallelism at application level

# GPU-VSIPL Implementation

- Full, compliant implementation of VSIPL Core-Lite Profile
- Fully encapsulated CUDA backend
  - Leverages CUFFT library
  - All VSIPL functions accelerated
- Core Lite Profile:
  - Single precision floating point, some basic integer
  - Vector & Sxalar, complex & real support
  - Basic elementwise, FFT, FIR, histogram, RNG, support
  - Full list: <http://www.vsipl.org/coreliteprofile.pdf>
- Also, some matrix support, including `vsip_fftm_f`

# CUDA Programming & Optimization

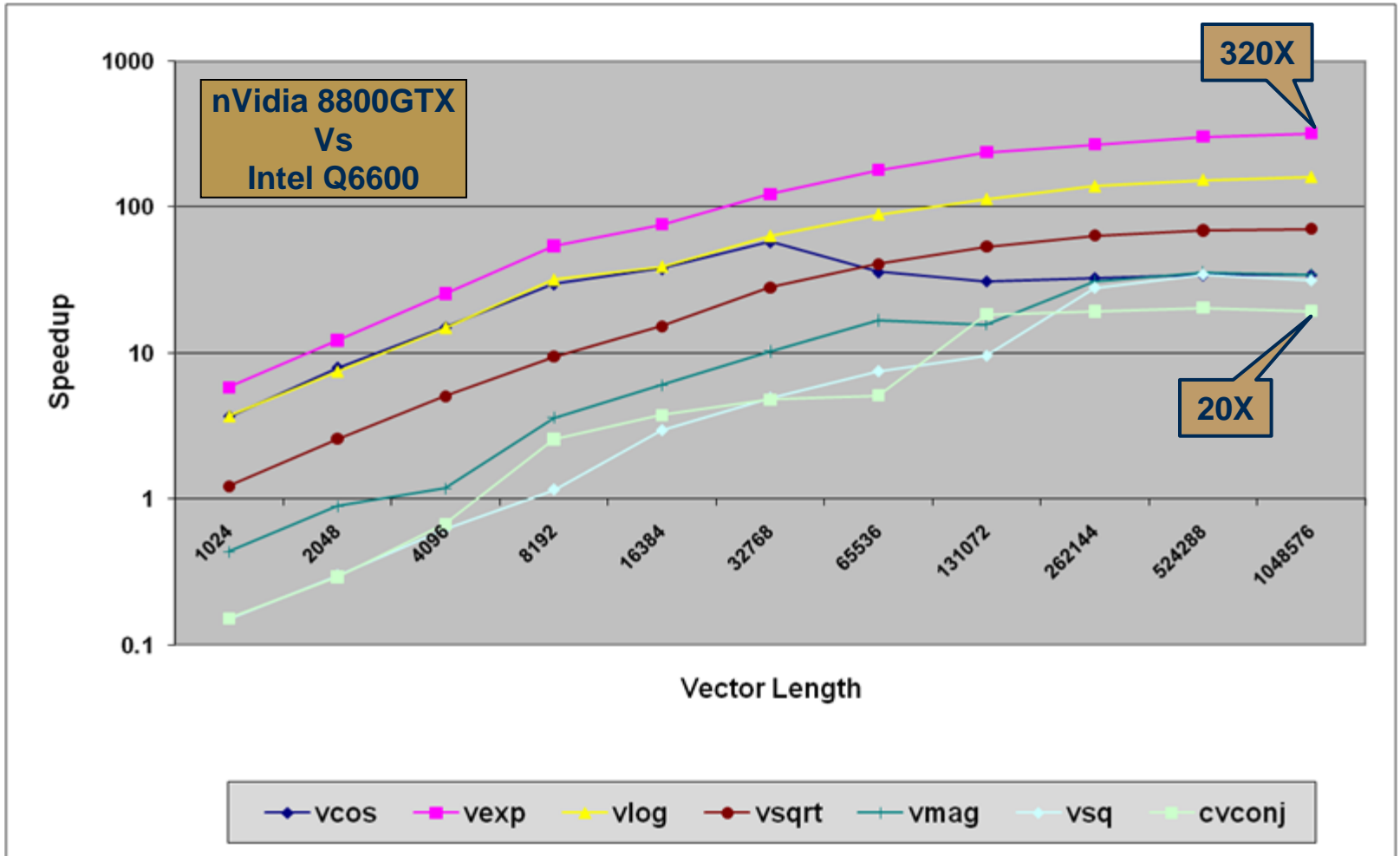
## CUDA Programming Model



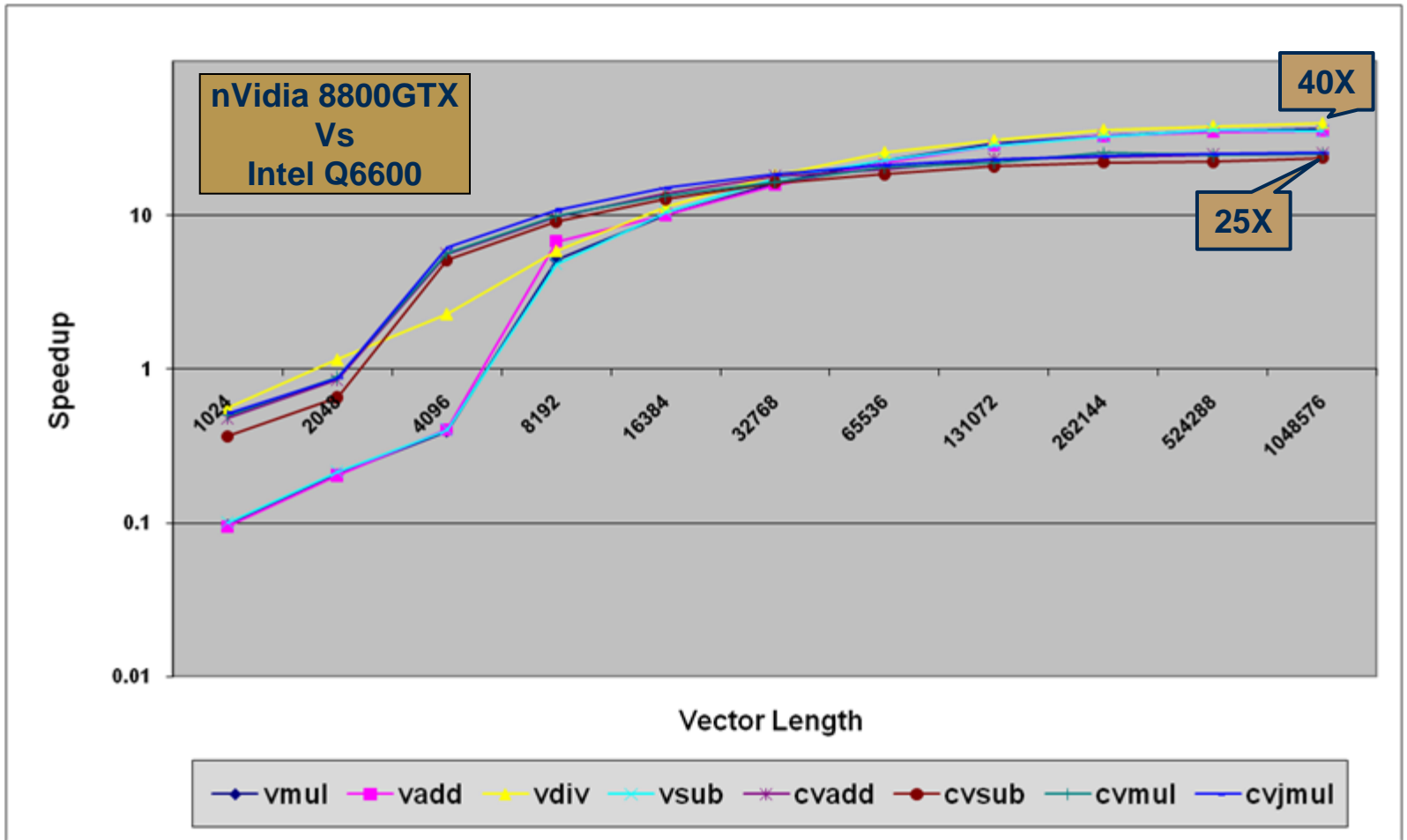
## CUDA Optimization Considerations

- Maximize occupancy to hide memory latency
  - Keep lots of threads in flight
  - Carefully manage memory access to allow coalesce & avoid conflicts
  - Avoid slow operations (e.g integer multiply for indexing)
  - Minimize synch barriers
  - Careful loop unrolling
  - Hoist loop invariants
  - Reduce register use for greater occupancy
- “*GPU Performance Assessment with the HPEC Challenge*” – Thursday PM

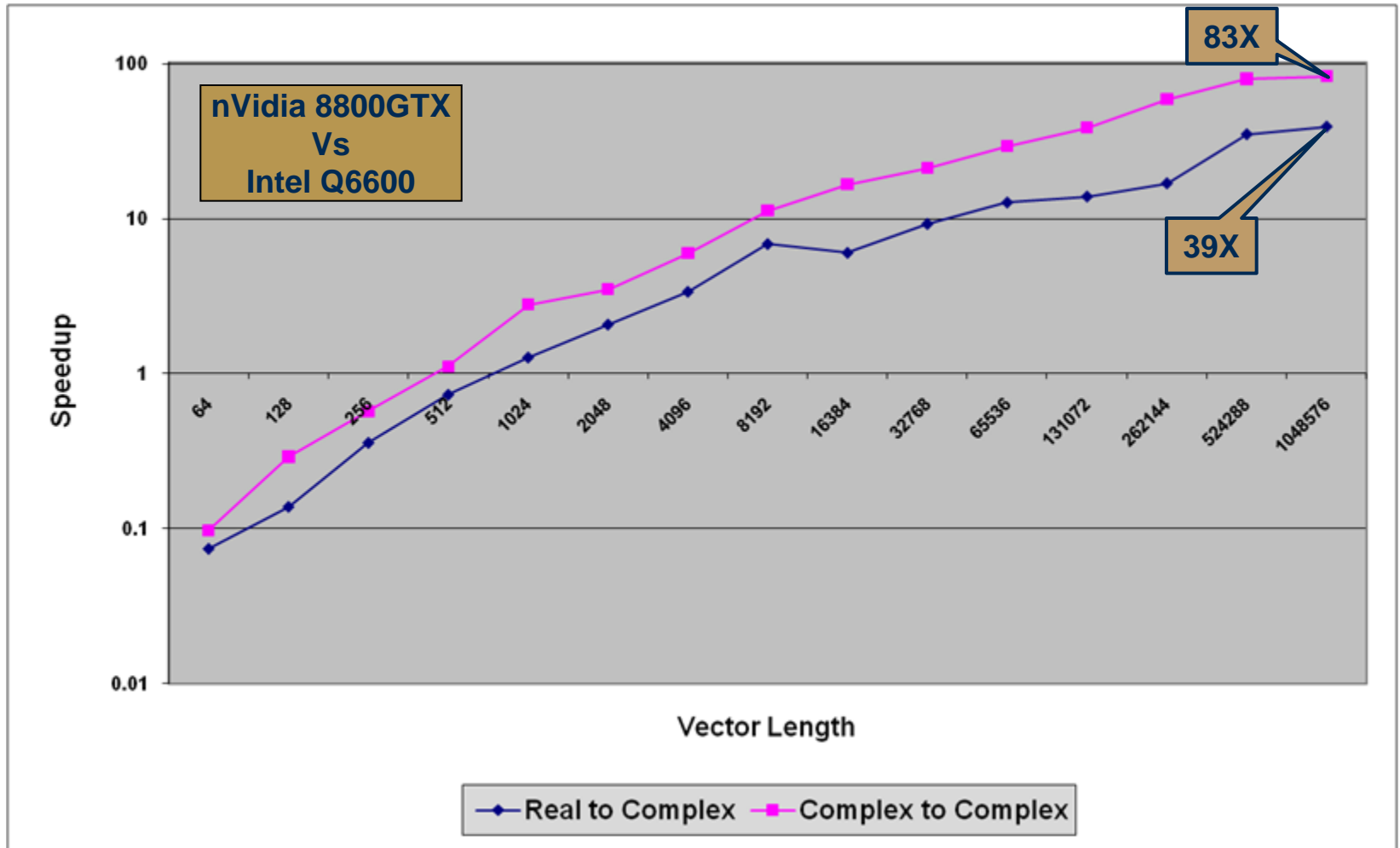
# GPU VS IPL Speedup: Unary



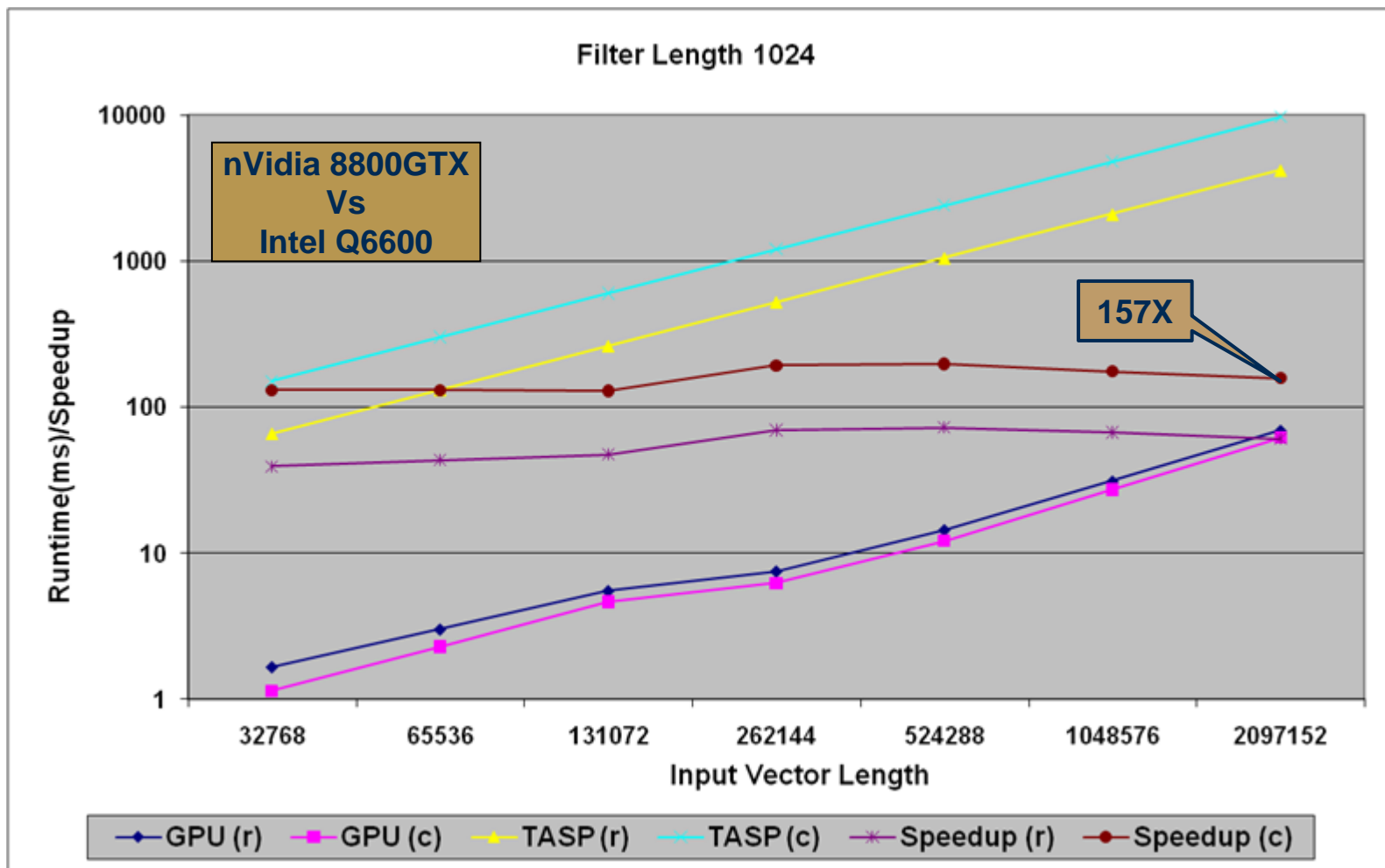
# GPU VS IPL Speedup: Binary



# GPU VS IPL Speedup: FFT



# GPU VSIPL Speedup: FIR



# Application Example: Range Doppler Map

- Simple Range/Doppler data visualization application demo
- Intro app for new VSIPPL programmer
- 59x Speedup TASP → GPU-VSIPPL
- No changes to source code

Section	<u>9800GX2</u> Time (ms)	<u>Q6600</u> Time (ms)	<u>Speedup</u>
Admit	8.88	0	0
Baseband	67.77	1872.3	28
Zeropad	23.18	110.71	5
Fast time FFT	47.25	5696.3	121
Multiply	8.11	33.92	4
Fast Time FFT <sup>-1</sup>	48.59	5729.04	118
Slow time FFT, 2x CT	12.89	3387	263
$\log_{10}  . ^2$	22.2	470.15	21
Release	54.65	0	0
<b>Total:</b>	<b>293.52</b>	<b>17299.42</b>	<b>59</b>

# GPU-VSIPL: Future Plans

- **Expand matrix support**
- **Move toward full Core Profile**
- **More linear algebra/solvers**
- **VSIPL++**
- **Double precision support**

# Conclusions

- GPUs are fast, cheap signal processors
- VSIPL is a portable, intuitive means to exploit GPUs
- GPU-VSIPL allows easy access to GPU performance without becoming an expert CUDA/GPU programmer
- 10-100x speed improvement possible with no code change
  
- Not yet released, but unsupported previews may show up at:  
<http://gpu-vsimpl.gtri.gatech.edu>