Tapping the TeraFLOP Potential of GP-GPU for High-Performance Computing Applications

Dr. Brooks Moses,

Sourcerer, Mentor Graphics

Dr. Gil Ettinger,

Consultant, Sensor Exploitation R&D

Eran Strod,

Curtiss-Wright Embedded Computing

IEEE

Advancing Technology for Humanity

Outline

- **GPU Overview and Benchmarks**

  Dr. Brooks Moses, Sourcerer, Mentor Graphics

- **Image Processing Algorithms**

  Dr. Gil Ettinger, Consultant, Sensor Exploitation R&D

- **Hardware for Embedded GPUs**

  Eran Strod, System Architect, Curtiss-Wright

GPU Overview and Benchmarks

Brooks Moses

Mentor Graphics

Objective

What kind of performance can I expect from a GPU, and what do I have to do to get it?

Background Application

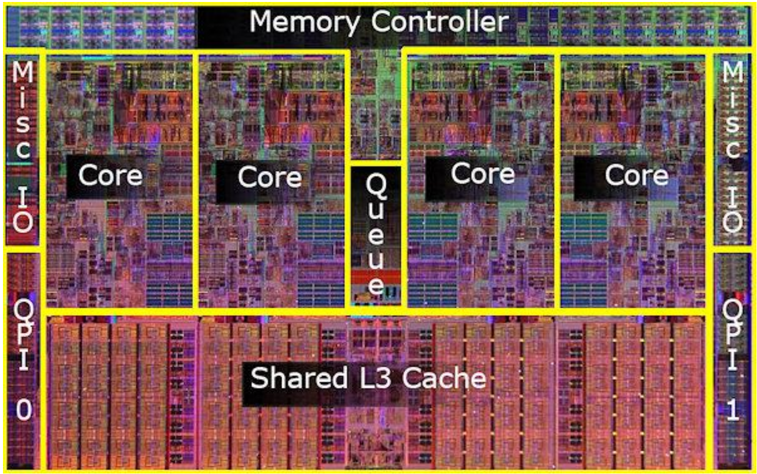
Mentor Embedded's **Sourcery VSIPL++** library

- Cross-platform API for signal, vector, and image processing.
- Encapsulated data model (handles storage, locality, etc.)

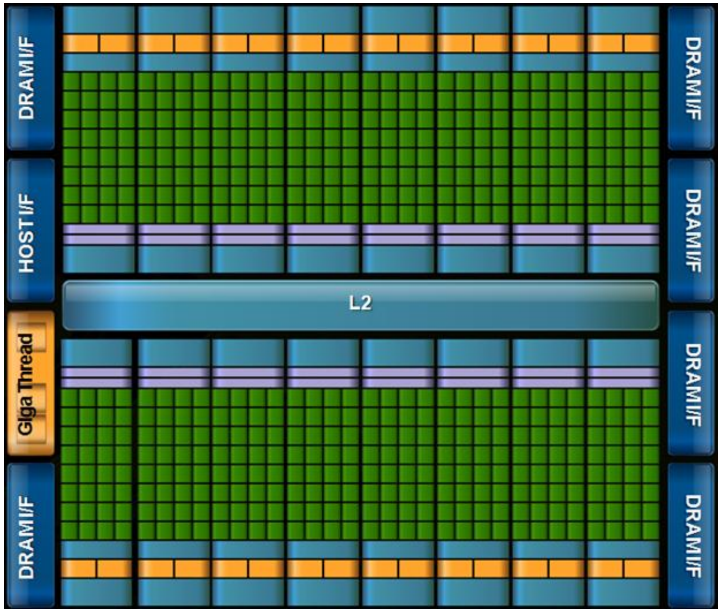
Development questions in producing our Sourcery VSIPL++ for NVIDIA CUDA port:

- What functions execute on the GPU, and when?
- How do we manage data locations for best performance?
- How do we make this all transparent to the user?

GPU Architecture



Intel Core i7 CPU
(Source: Intel)



NVIDIA Fermi GPU
(Source: NVIDIA)

- Four independent cores

- Sixteen 32-core “streaming multiprocessors”

GPU Architecture

- Each Streaming Multiprocessor (SM) has one instruction decoder for all 32 cores.
  - Thus: Groups of 32 threads (called “Warps”) execute in lockstep.
- SMs use hardware multithreading to overlap multiple warps and hide latency.
  - Sets of Warps on an SM (called “Blocks”) can share local memory.
  - Can execute hundreds of Blocks simultaneously.
  - Limited number of programs (called “Kernels”) executing concurrently: 4 on Fermi, 1 on older GPUs.

NVIDIA CUDA

CUDA ("Compute Unified Device Architecture"):

- An API for general-purpose GPU programming
- NVIDIA proprietary solution, but very popular
  - Main competitor is OpenCL.
- Includes a C-like language for writing GPU kernels

Comparing CPU and GPU Performance

Processors used for comparison:

x86: Intel Core i7 (Nehalem), 4 cores (hyperthreaded), 3.2 GHz, peak performance of 102.4 GFLOPS/s.

CUDA: NVIDIA Tesla C1060, 240 cores, 1.3 GHz, peak performance of 933 GFLOPS/s.

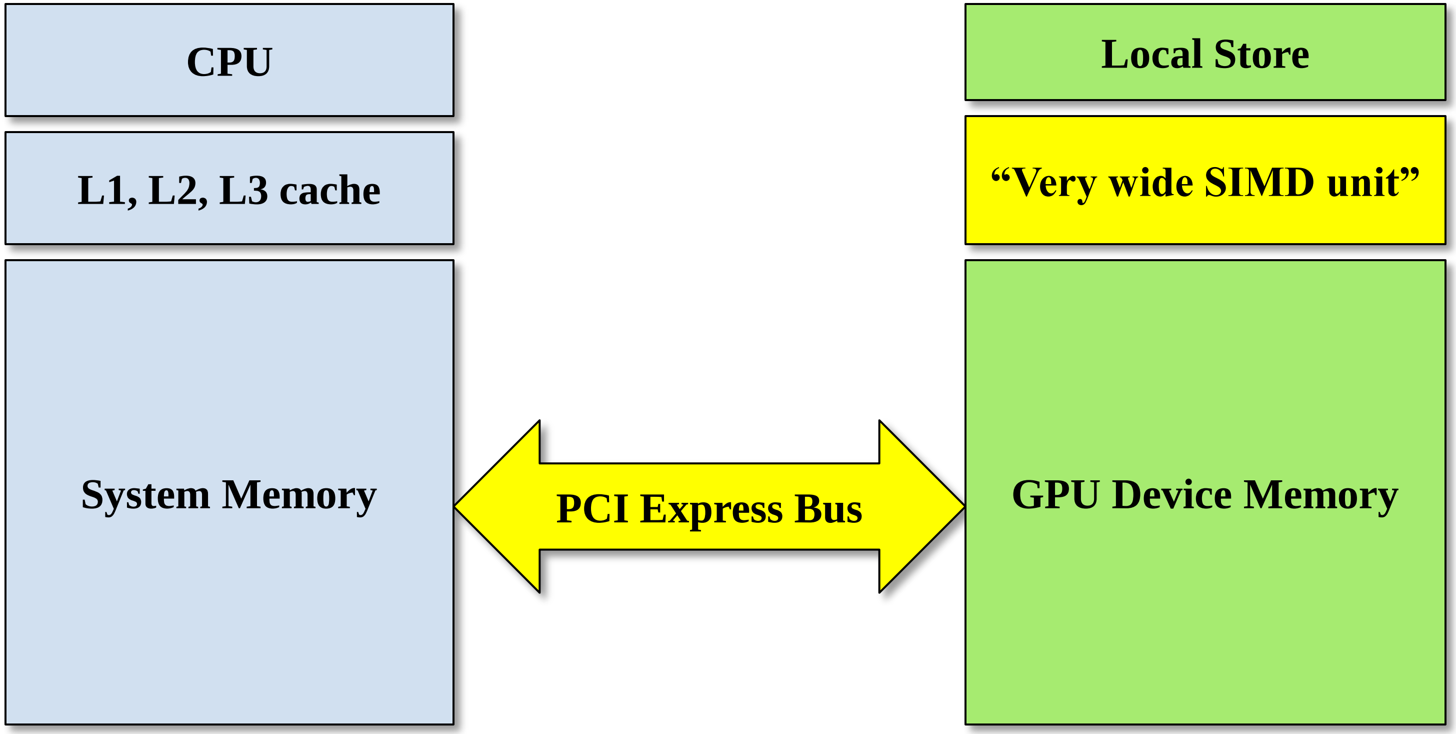
(GFLOP/s = 10^9 floating-point operation per second)

Note: This is previous-generation technology; current versions of both are about 2x faster.

Comparing CPU and GPU Performance

- Benchmarking program choice is important!
- Many GPU benchmark results show more than 100x performance improvement.
  - With only 10x more GFLOP/s, is this believable?
  - Typically, this is a comparison to an unoptimized, non-vectorized, single-threaded CPU implementation.
- Realistic comparisons require high-quality implementations on ***both*** CPU and GPU.
  - E.g., Intel’s IPP library vs. NVIDIA’s CUBLAS library.

Oversimplified model of a GPU



- GPU cores are effectively a "SIMD unit" (Single Instruction, Multiple Data) with flexible data width, masking, etc.
- PCI Express bus and GPU cores are key pieces for understanding GPU performance.

Performance

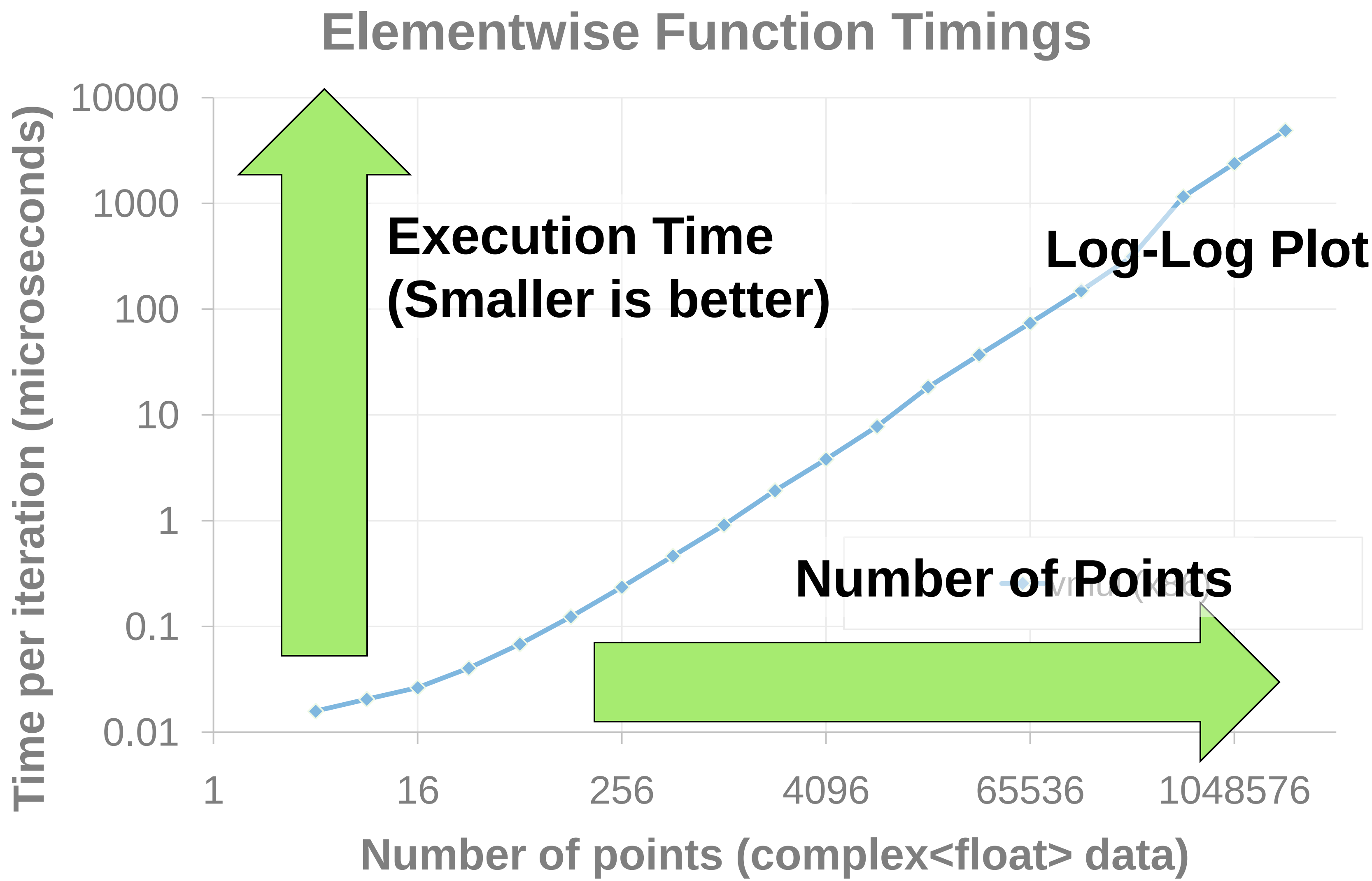
What does the performance of this GPU-core “SIMD unit” look like?

Best performance cases:

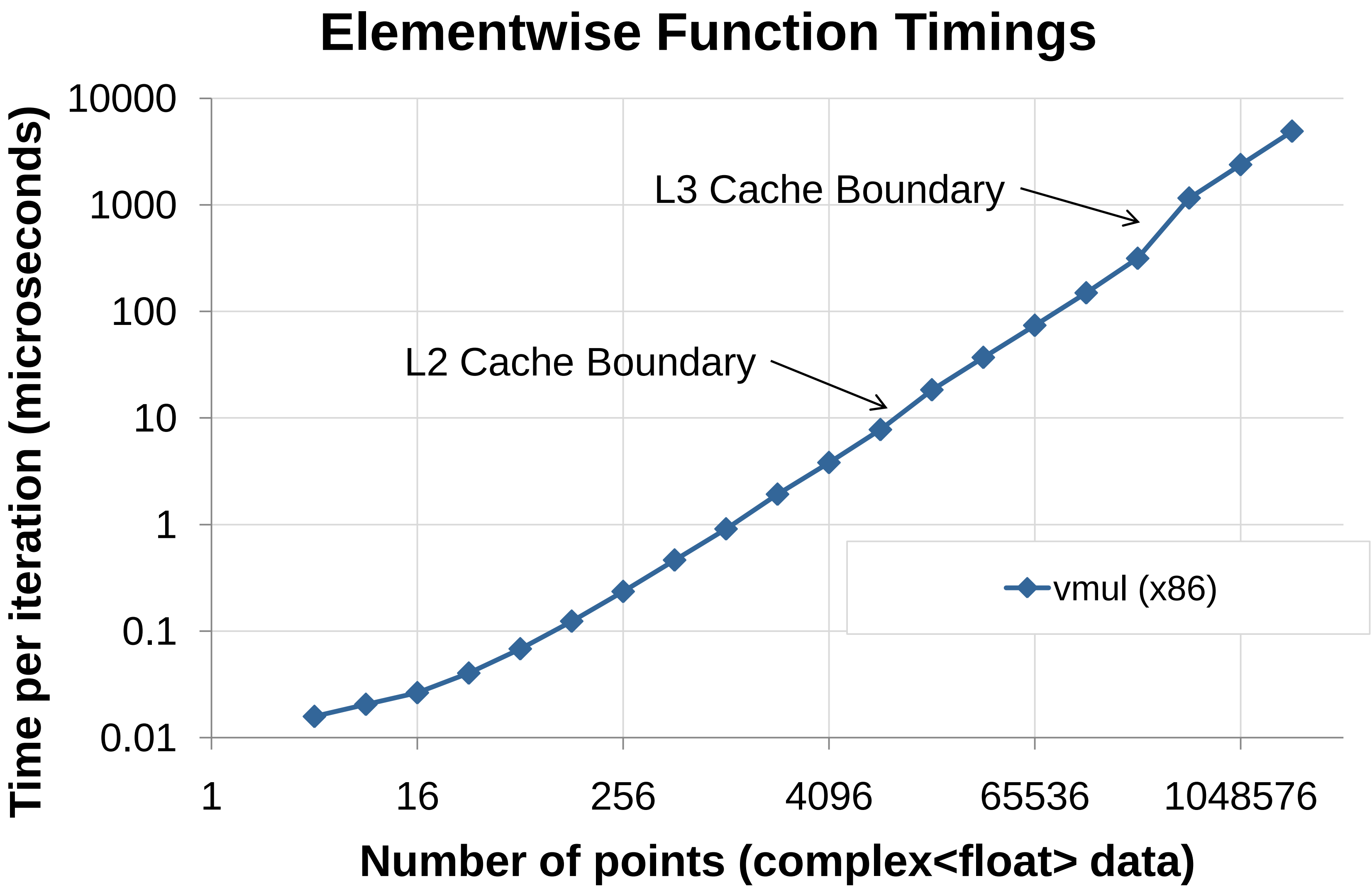
- Large amounts of data
- Every element of data touched
- Identical operations (with masking) on each piece of data.

Start with a simple example: A = B * C (with vectors)

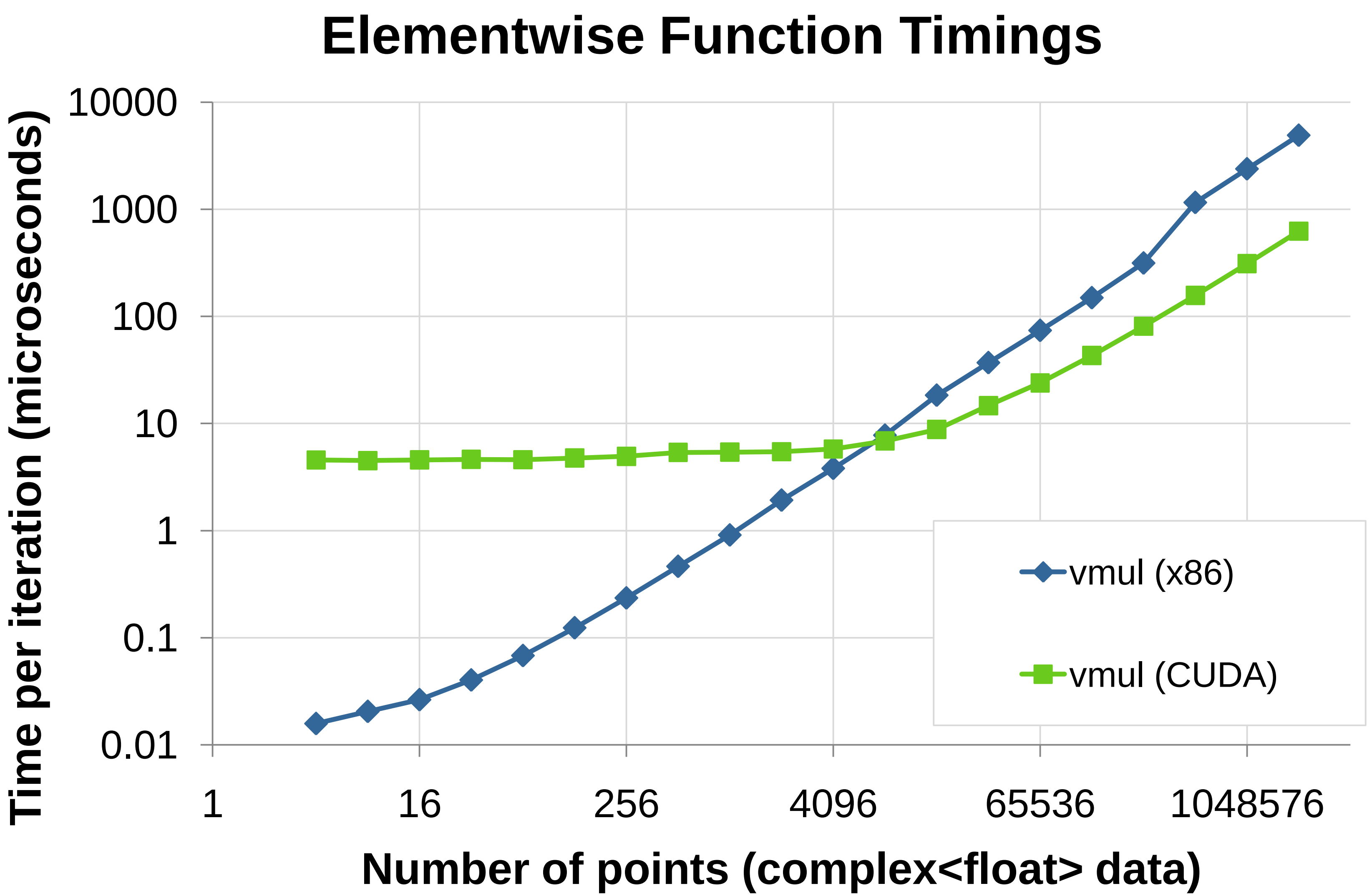
Elementwise Vector Operations



Elementwise Vector Operations



Elementwise Vector Operations



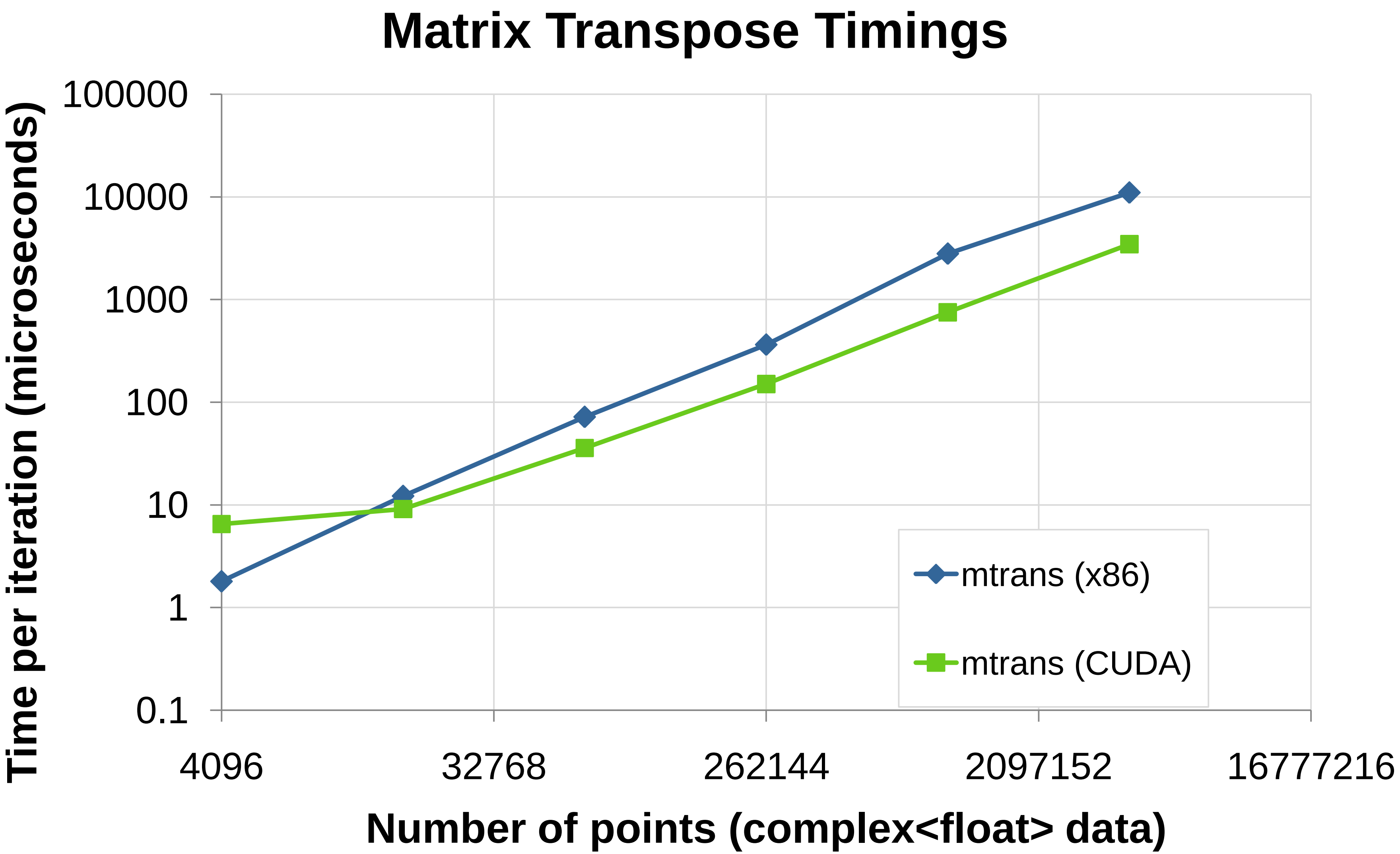
Performance

Key observations:

- Large fixed cost (~5 microseconds) for initiating an operation and executing one instruction.
- Above 16k points, execution time is proportional to size.
- GPU is faster with 8k or more points.
- Best performance is about 10x the CPU performance.

These results are typical for other, more complicated operations that fit the same criteria.

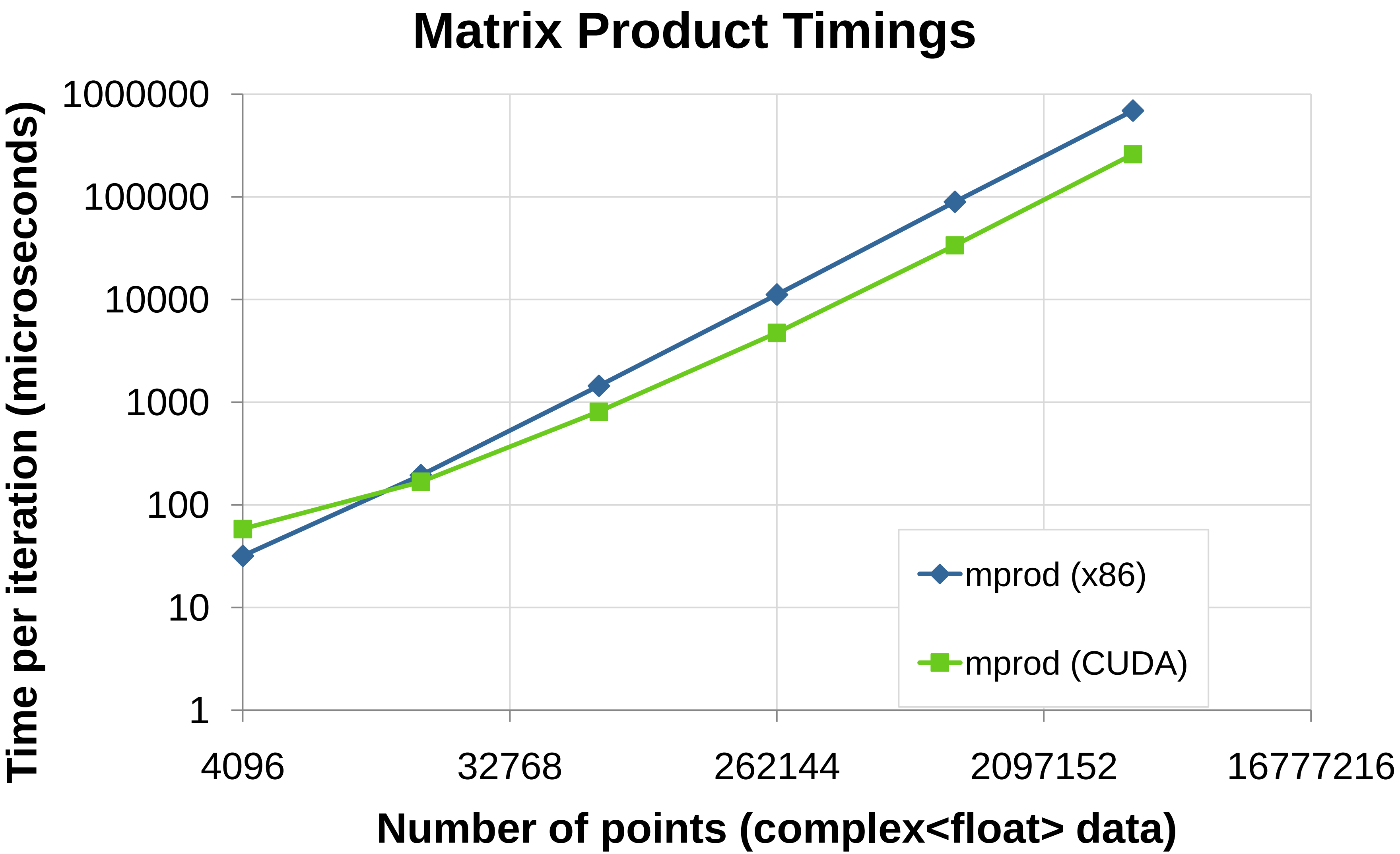
Matrix Transposition



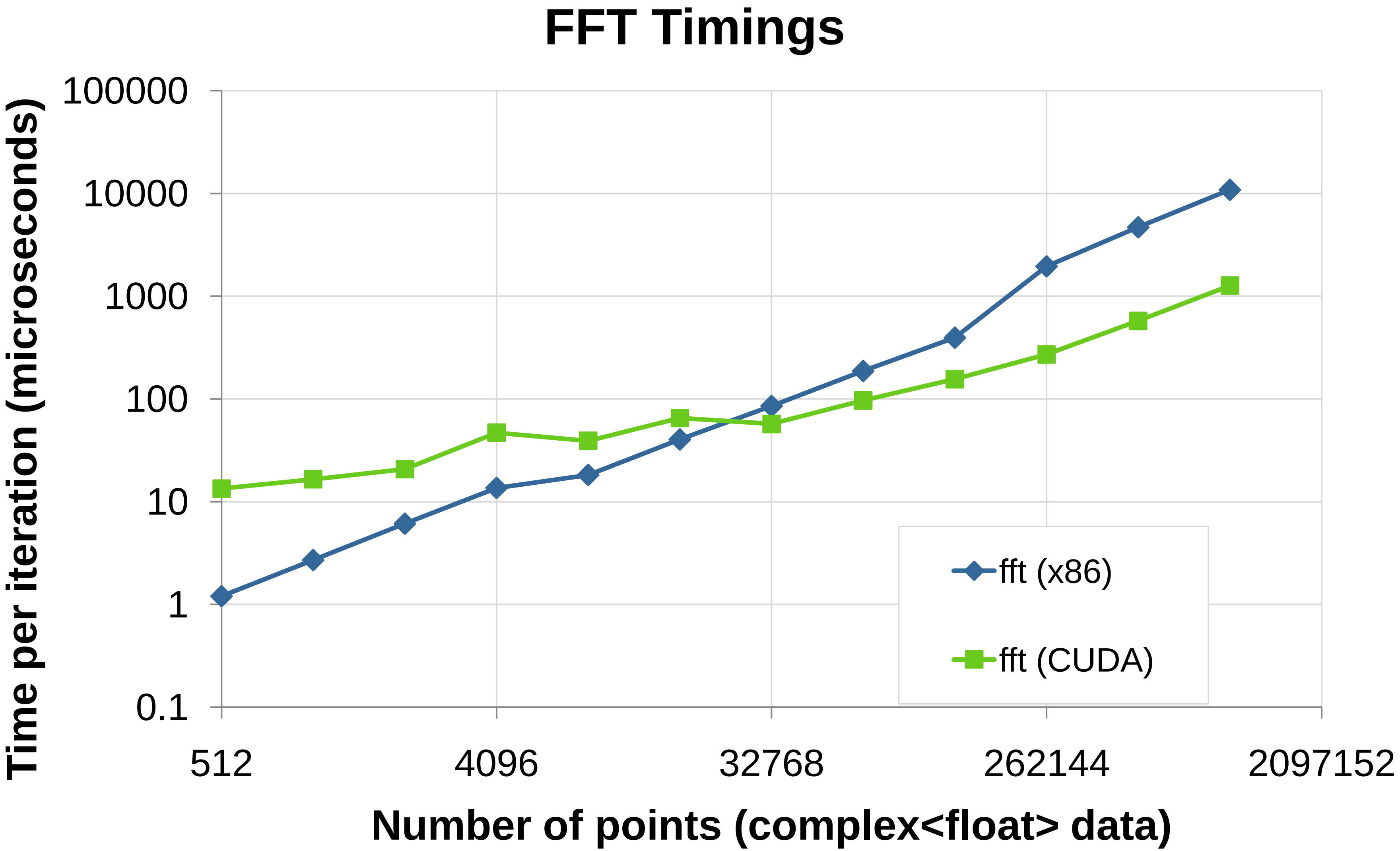
CodeSourcery

mentor embedded

Matrix Product



Fast Fourier Transform



System and Device Memory

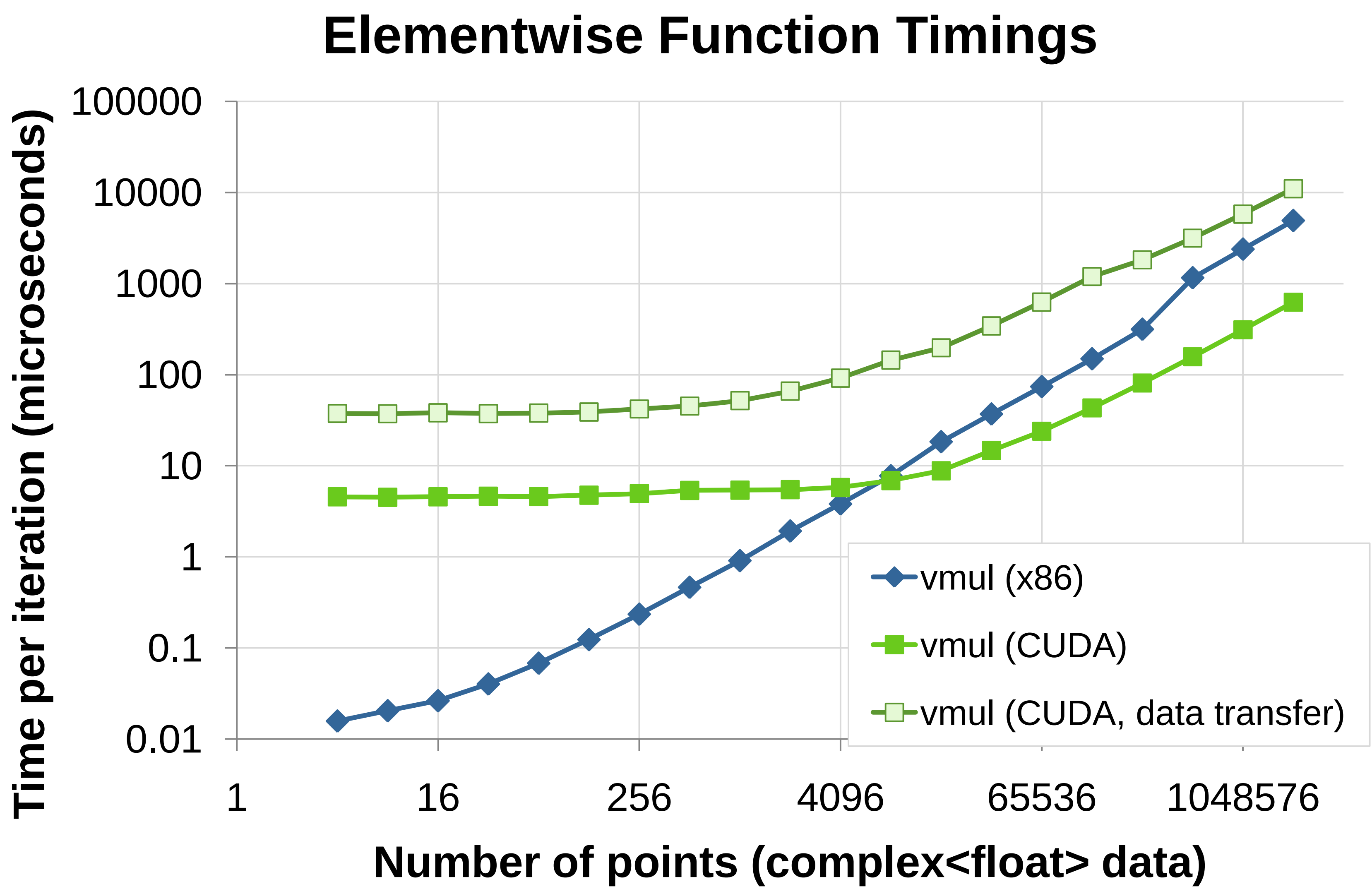
Memory structure affects performance:

- CPU can only use data from system memory
- GPU can only use data from device memory
- Data is transferred between them via the PCI-E bus

How does this need for data transfer affect performance?

Back to the A = B * C example.

Elementwise Vector Operations



Conclusions and Recommendations

Performance of GPUs varies with algorithm, data size

What works well on a GPU?

- Algorithms with large SIMD-like operations
  - Data sizes greater than 4k elements
  - Same operations (with masking) executed on all elements

Expected best performance: Typically 3x to 10x faster than CPU on algorithms that work well on GPU.

Conclusions and Recommendations

Data transfers from CPU to GPU are costly.

For optimal performance, minimize data transfers:

- Large blocks of algorithm should be executed entirely on the GPU:
    - Thus, need GPU implementations of all operations within that block of algorithm. *Not just core inner loops!*
    - Use libraries to minimize development time.
    - NVIDIA, CuBLAS, CuFFT, Thrust, etc.; CULAtools.
    - Sourcery VSIPL++ provides portable wrapper around all of these.

Sourcery VSIPL++

How does Sourcery VSIPL++ make this easy?

Separation of algorithm and implementation:

- Encapsulated data objects
  - Move data between CPU and GPU automatically as needed.
  - Provides logging of when tranfers occur and transfer time.
- Portable function call syntax
  - Wraps best-of-class CUDA, CULA libraries and CPU libraries, along with providing additional operations.
  - Functions execute on CPU or GPU for best performance depending on data size and current location.

Image Processing for Video-based Scene Understanding

Opportunities for GP-GPU Parallelization of Image Processing Operations

June 15, 2011

Gil Ettinger

ettinger@alum.mit.edu

Image Processing for Video-based Scene Understanding

Image Processing Challenges:

- Detect and geo-locate movers – even if very small
- Maintain ID on movers – even if moving slowly, in dense traffic, or partially occluded
- Filter spurious motion – such as smoke or natural clutter
- Interpret complex scenes with wide range of stationary and moving objects
- Automate processing for real-time exploitation of high bandwidth data streams

Generalized Processing Flow for Video-based Scene Understanding

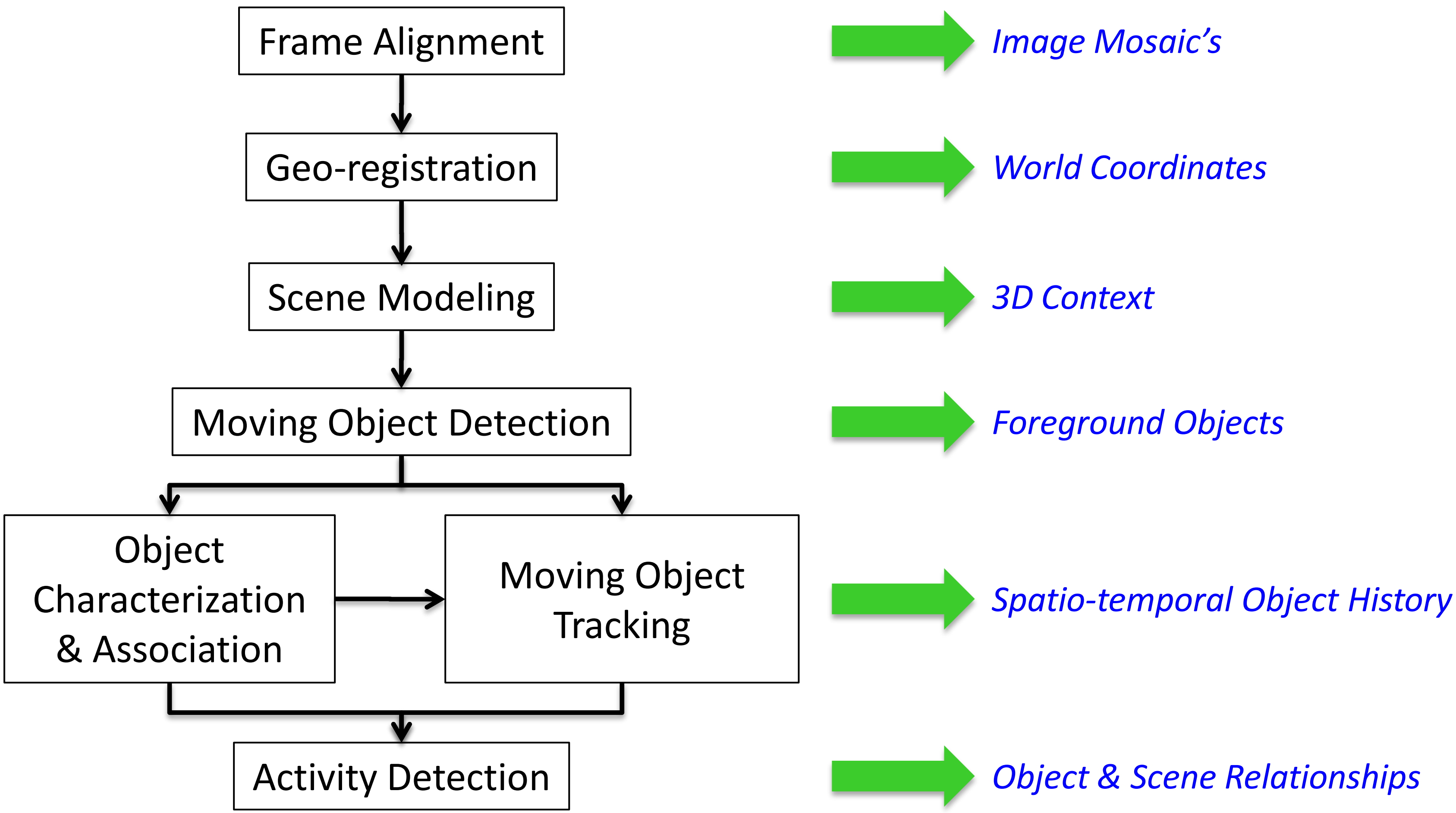


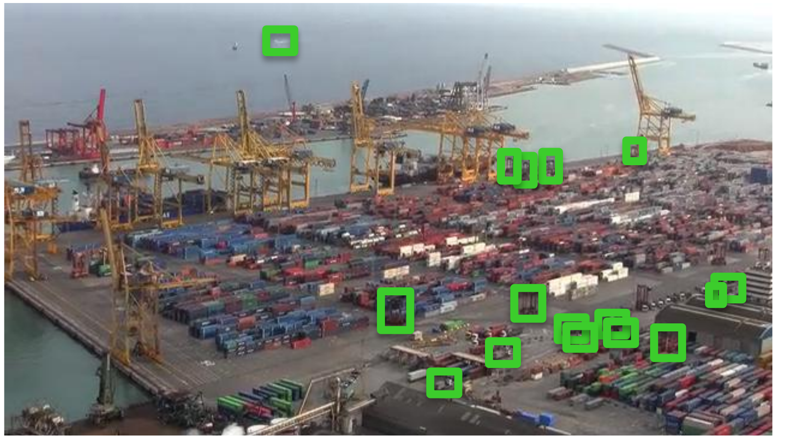
Image Processing Algorithms (2)

- <u>Geo-registration</u>:
  - Align (periodic) ortho-projected frames to reference ortho-image/map
  - Approaches:
    - Point Feature (e.g., corner (gradient-intensive)) Alignment
    - Edge Feature (gradient-intensive) Alignment
  - Computational Complexity:
    - Touch large number of pixels – subset of pixels in subset of frames
    - Transform search often performed hierarchically
    - Non-linear optimization requires less distributed processing

Image Processing Algorithms (3)

- Scene Modeling:
  - Extraction of 3D scene models (and other contextual information)
  - Approaches:
    - Shape from Shading/Texture (non-linear least-squares optimization)
    - Shape from Motion (surface/volume evolution via global photoconsistency/visibility optimization)
  - Computational Complexity:
    - Touch all pixels in subset of frames
    - Global optimization can be performed distributively on scene patches

Image Processing Algorithms (4)



- Moving Object Detection:
  - Identification of moving vehicles, people (and separation from other spurious motion)
  - Approaches:
    - Background Subtraction (statistical background modeling)
    - Optical Flow Segmentation (gradient feature matching)
  - Computational Complexity:
    - Touch all pixels in all frames
    - Reliable detection requires multi-frame analysis/learning
    - High degree of data parallelism

Image Processing Algorithms (5)

- Moving Object Tracking & Association:
  - Continuous maintenance of object ID through space and time
  - Approaches:
    - Kinematic Tracking:
      - Multiple Hypothesis Tracking (bounded search)
      - Particle Filtering (probabilistic modeling)
    - Object Appearance Association:
      - Intensity Correlation (sum of pixel intensity products)
      - Feature Association (gradient matching)
  - Computational Complexity:
    - Touch all detections in all frames
    - Number of associations (hypotheses) grows polynomially with detections
    - Reliable tracking requires multi-frame analysis and leveraging of site context
    - Tracking is generally a centralized process, but underlying object association functions are parallelizable

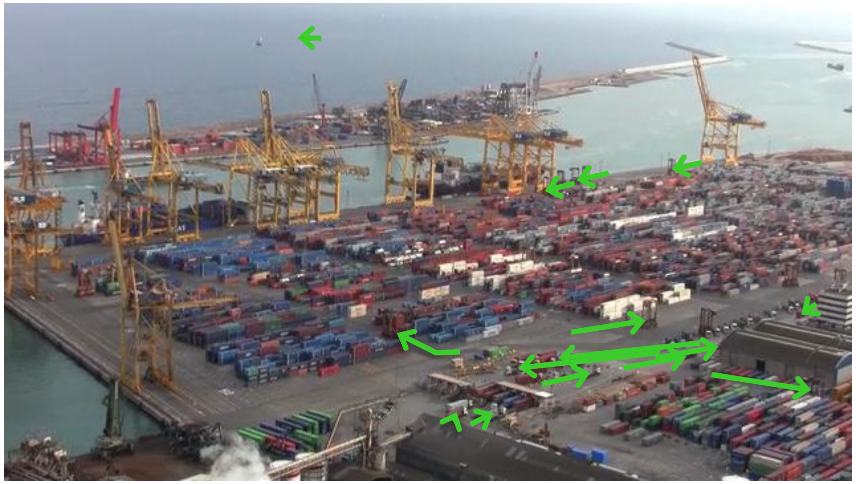


Image Processing Algorithms (6)

- Activity Detection:
  - Identify actions and events performed by individuals or groups of vehicles and/or people
  - Approaches:
    - Space-time Local Feature Trajectory Classification: Intensities, Gradients, Corners
    - Space-time Feature/Object Relationship Classification
    - Model-based Constrained Search
    - Multi-sensor Fusion: Audio, Multi-spectral, Multi-look
  - Computational Complexity:
    - Scene relationship finding requires touching most pixels, not just object detections and tracks
    - Feature extraction complexity is highly variable
    - Activity hypothesis search can involve search through high dimensional space

Deployment of GPUs

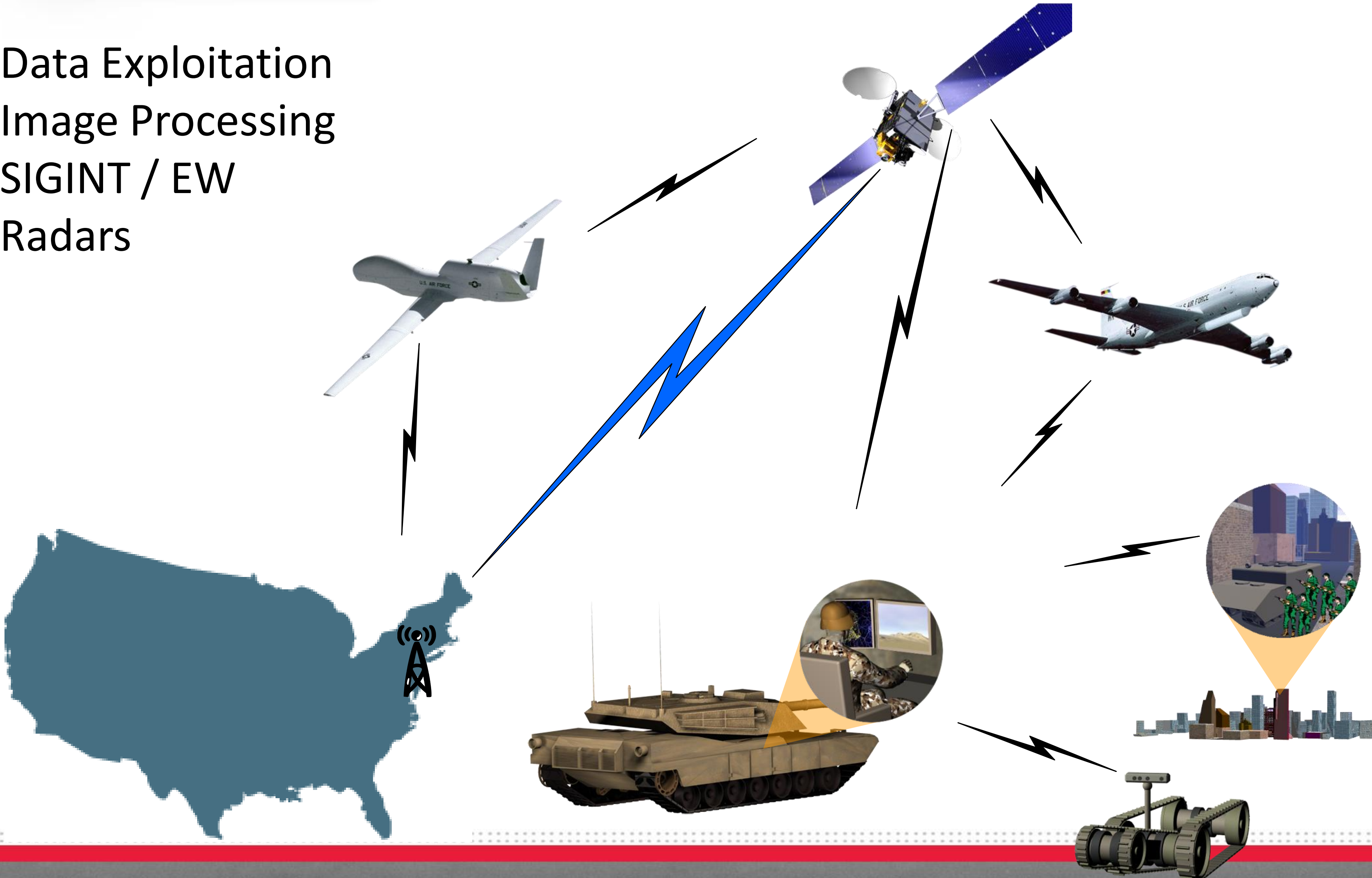
June 15, 2011

Eran Strod
eran.strod@curtisswright.com

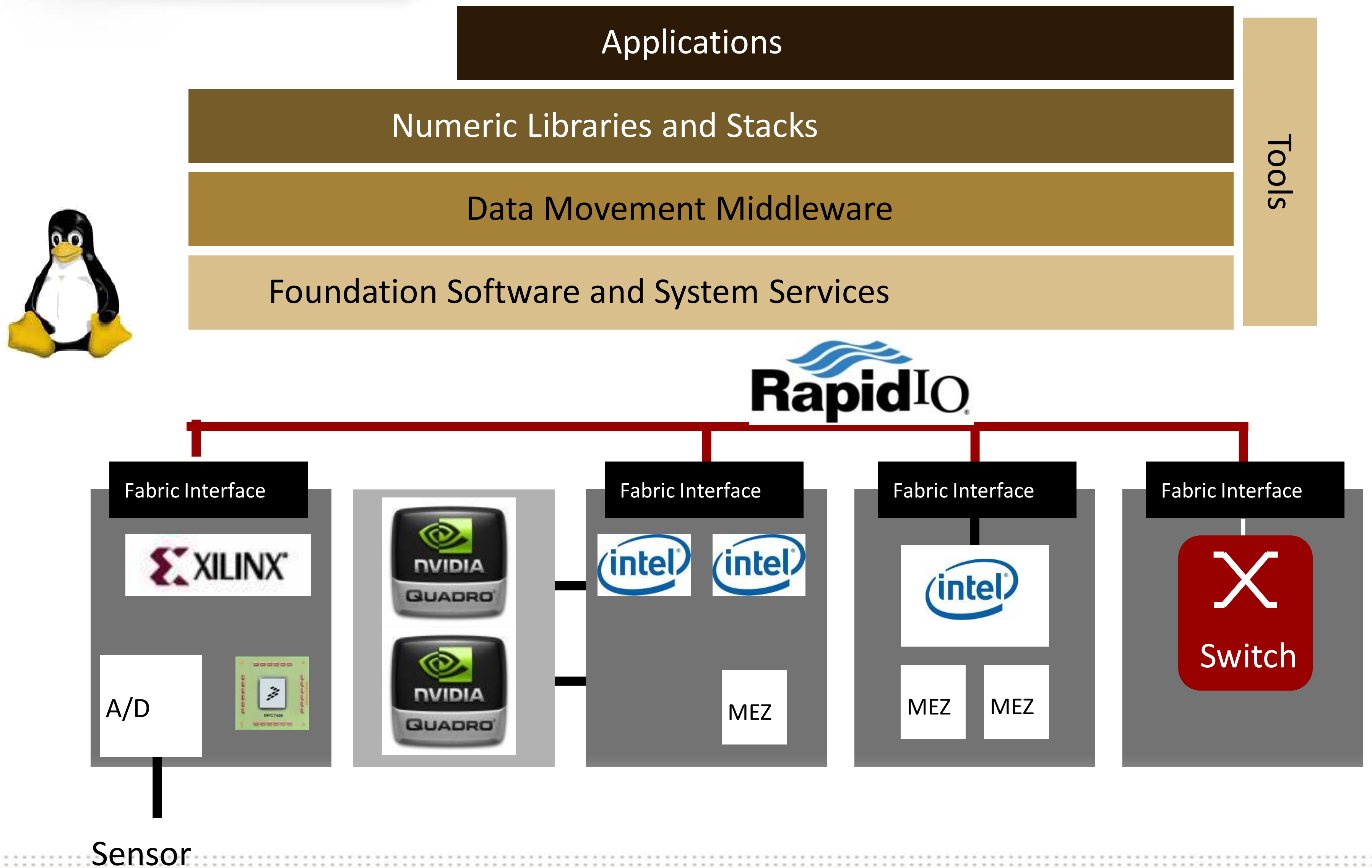
CURTISS WRIGHT Controls Embedded Computing

Target Deployments

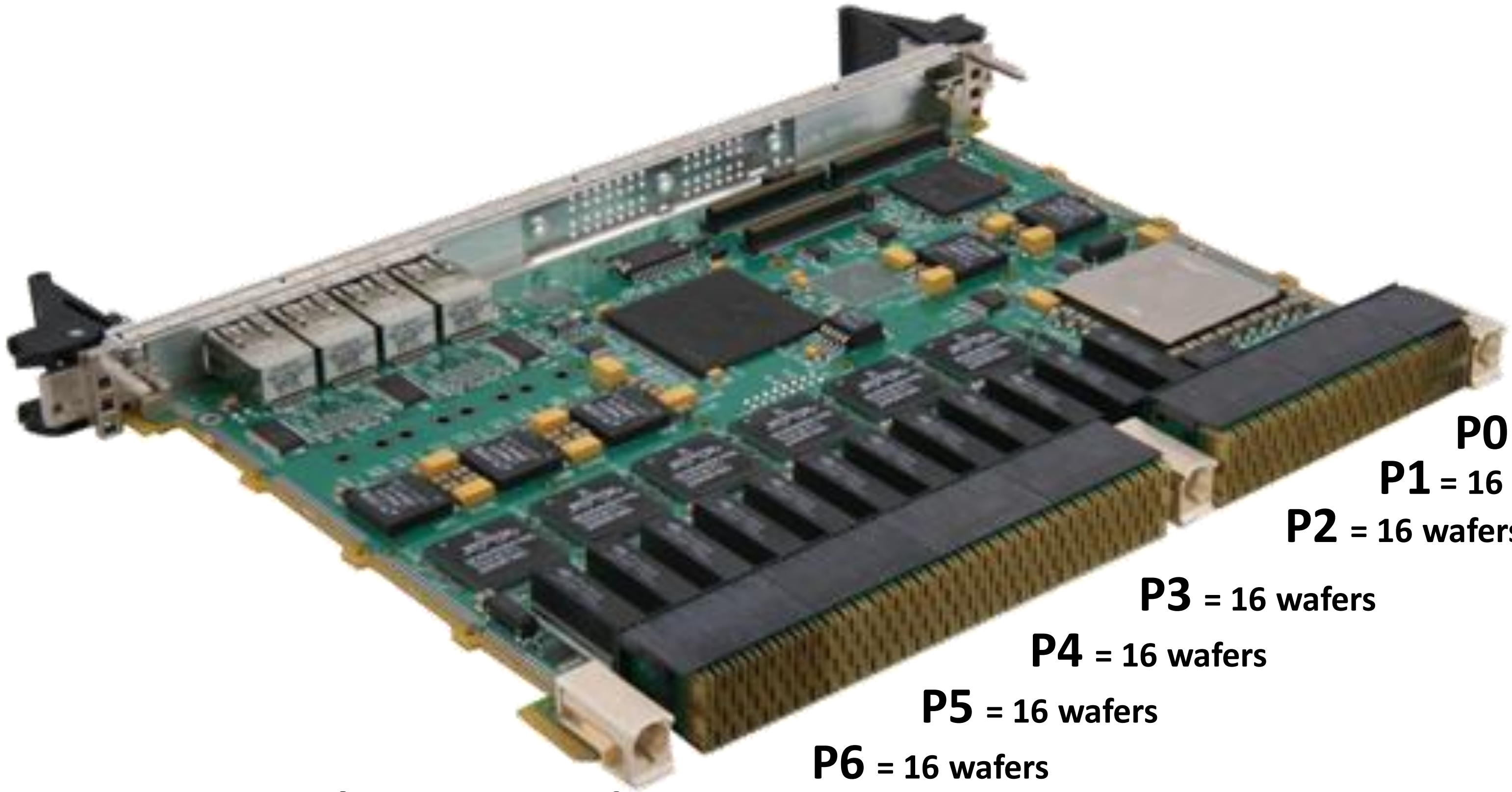
- Data Exploitation
- Image Processing
- SIGINT / EW
- Radars



Platform Architectures



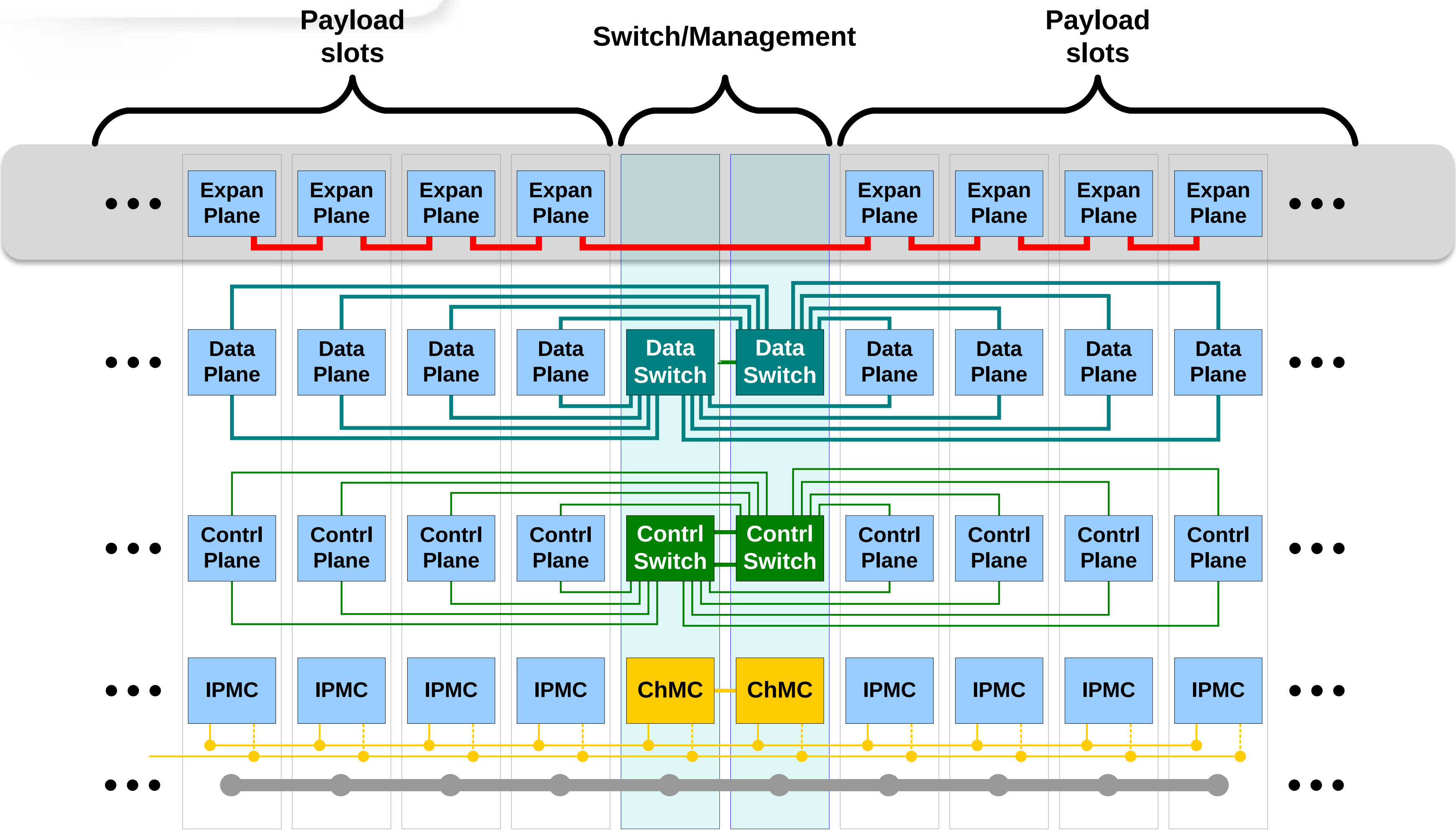
VPX Connectors



P0 = 8 wafers
P1 = 16 wafers
P2 = 16 wafers
P3 = 16 wafers
P4 = 16 wafers
P5 = 16 wafers
P6 = 16 wafers

Alignment and Keying Blocks (3) also provide safety ground

Expansion Plane

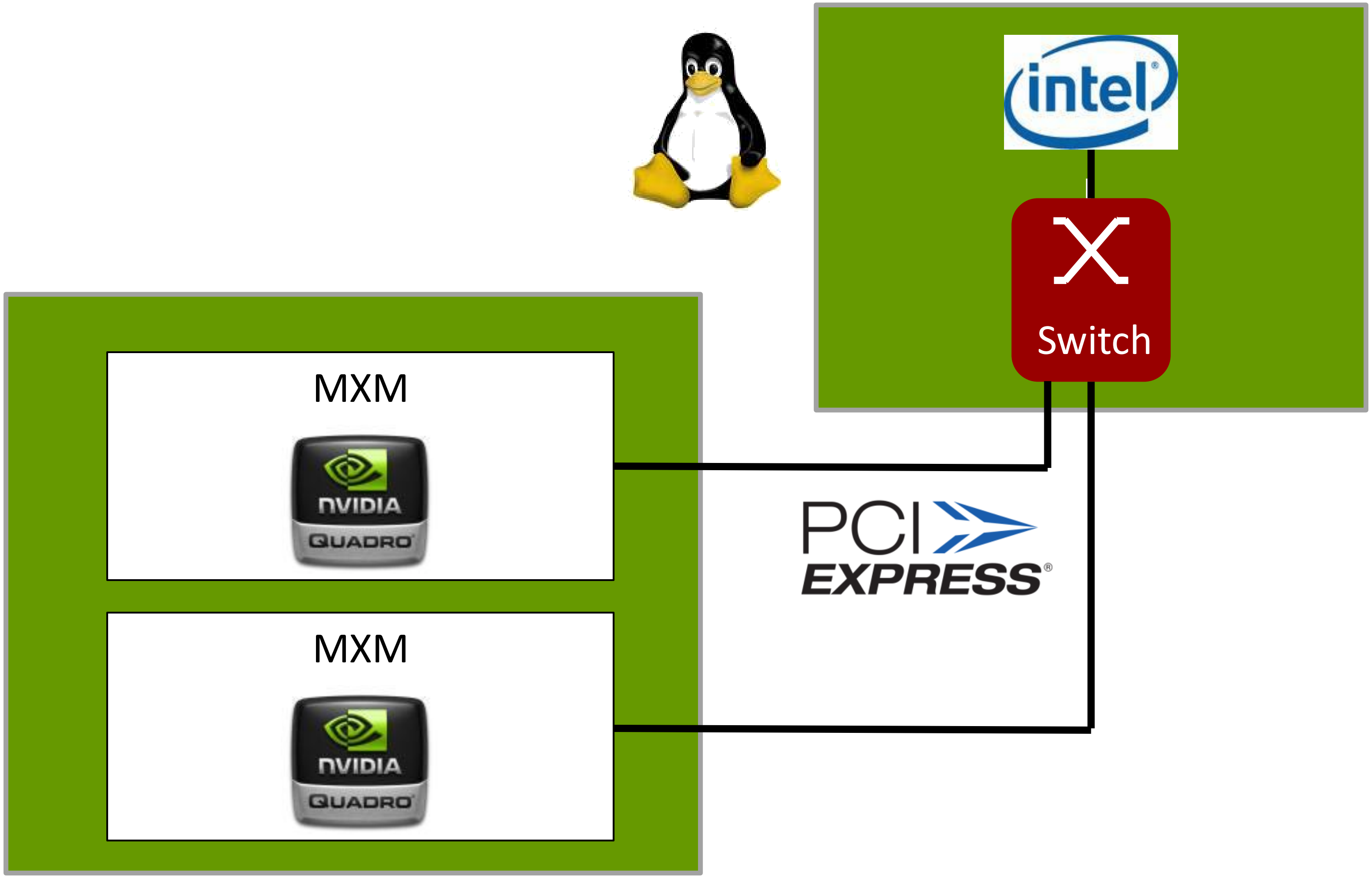


- Tightly coupled groups of boards and I/O
  - Typically PCI Express x8

About MXM

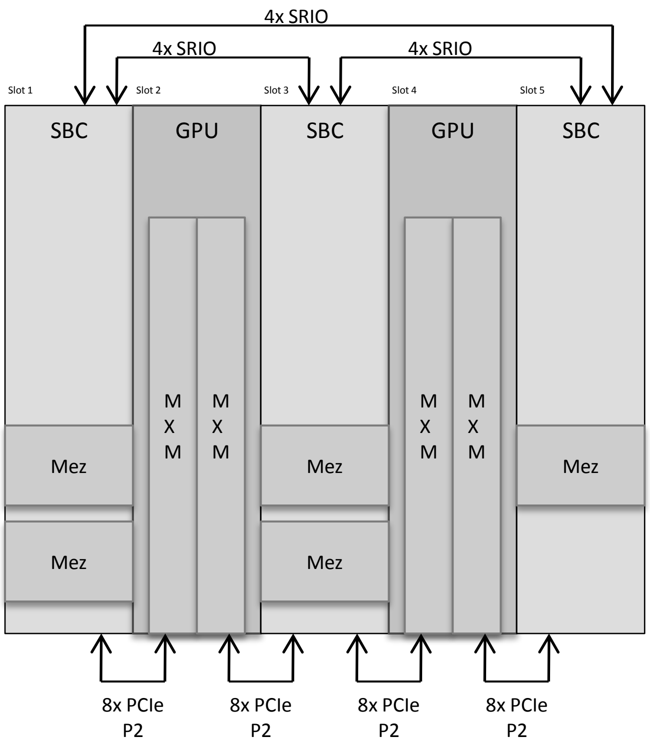
- MXM is the Mobile PCI Express Module. A standard form-factor for low-power, small form-factor applications
- Typical applications are laptop computers, blade and rack-mount servers.
- Thermal solution is customized for the end application
- Supports GPU devices up to approx 75W. Up to 16-lane PCIe
- Newest MXM version 3 type B modules
- http://www.mxm-sig.org/

Module Architecture

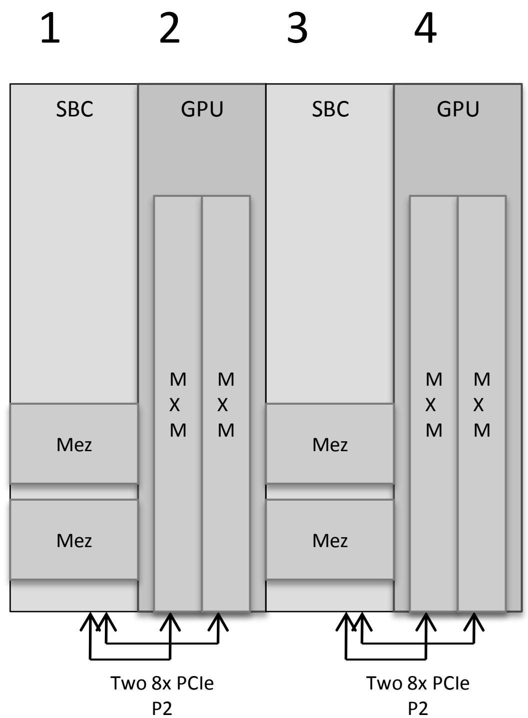


GPU Base Configurations

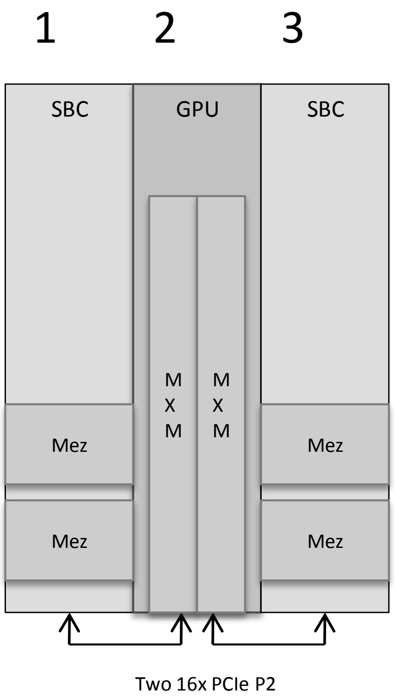
Data Plane Full Mesh



Standard Open VPX Expansion Plane



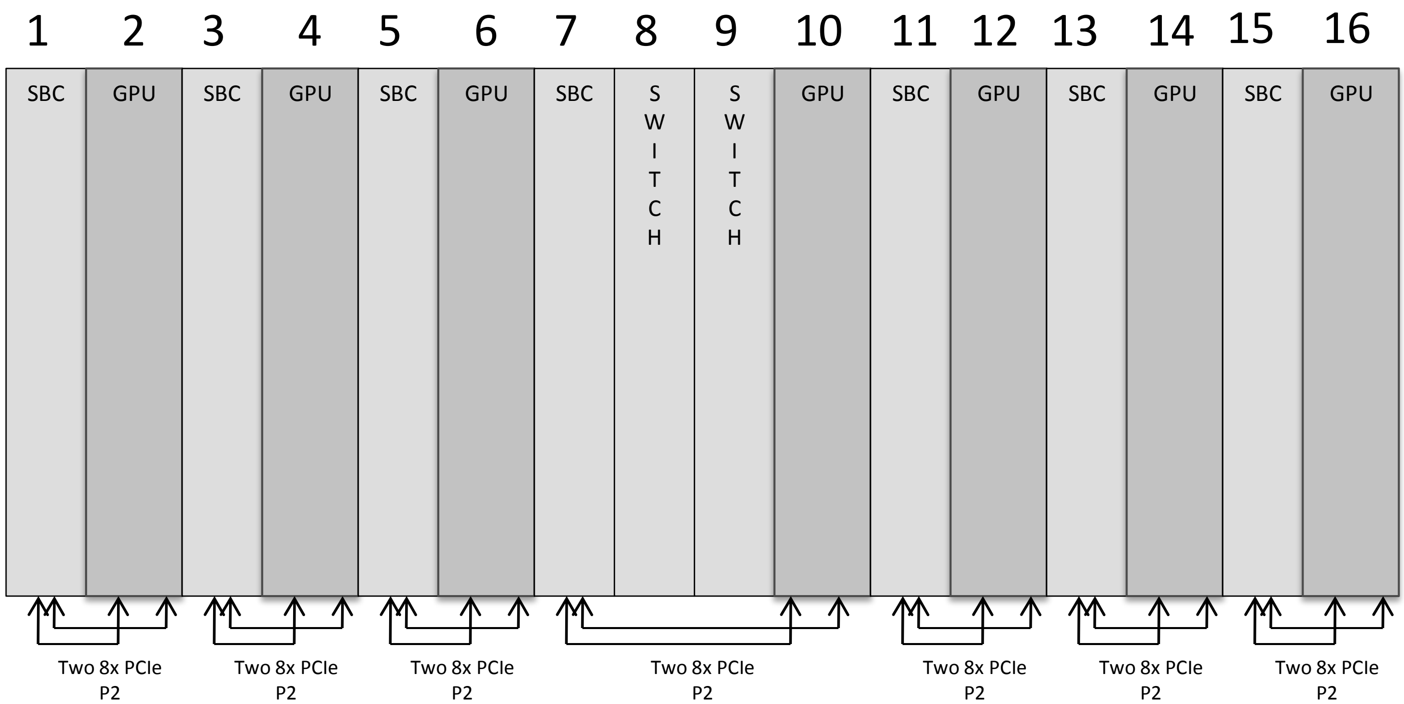
Expansion Plane
Two 8x PCIe Ports



Expansion Plane
Two 16x PCIe

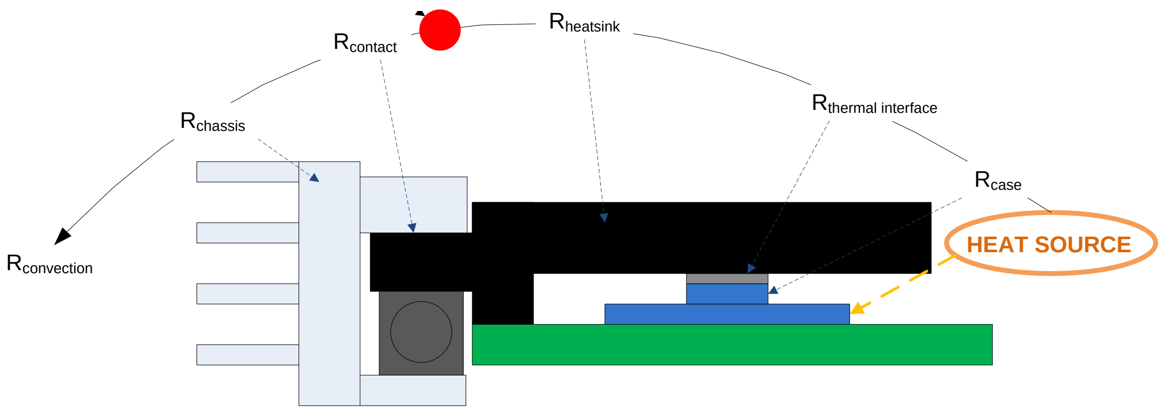
Scaling Up

Open VPX BKP6-CEN16: (14 Payload + 2 Switch)

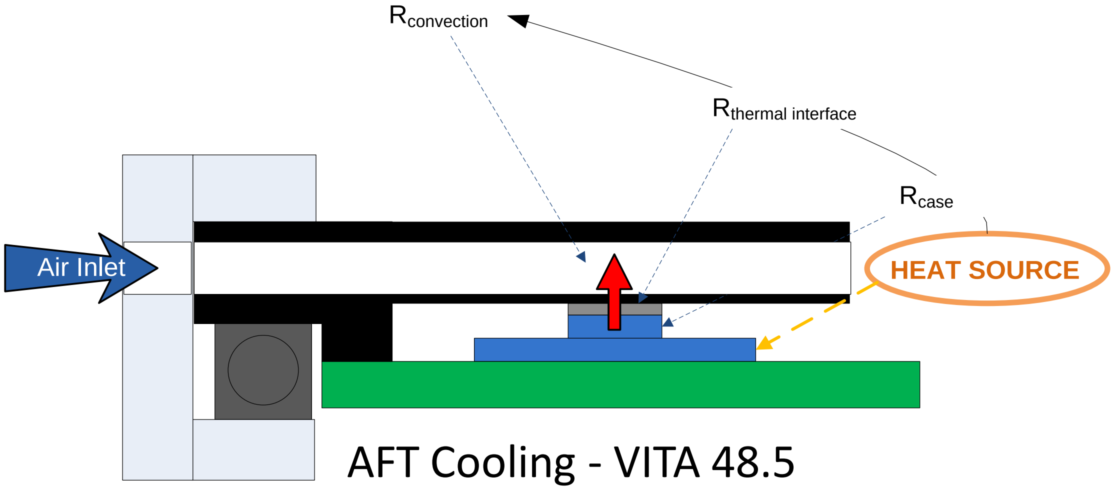


Expansion Plane
Two 8x PCIe Ports

AFT Cooling Heat Path

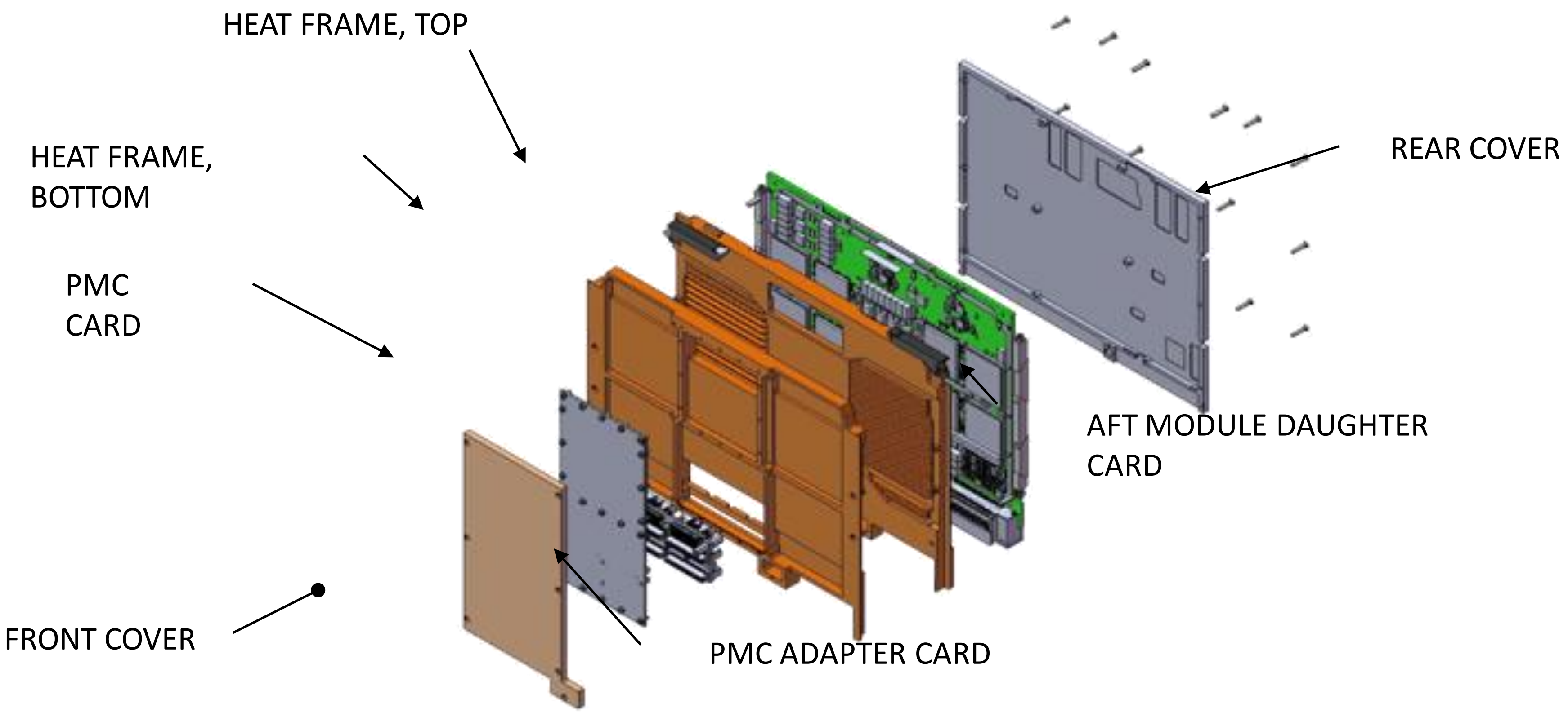


Conduction Cooling

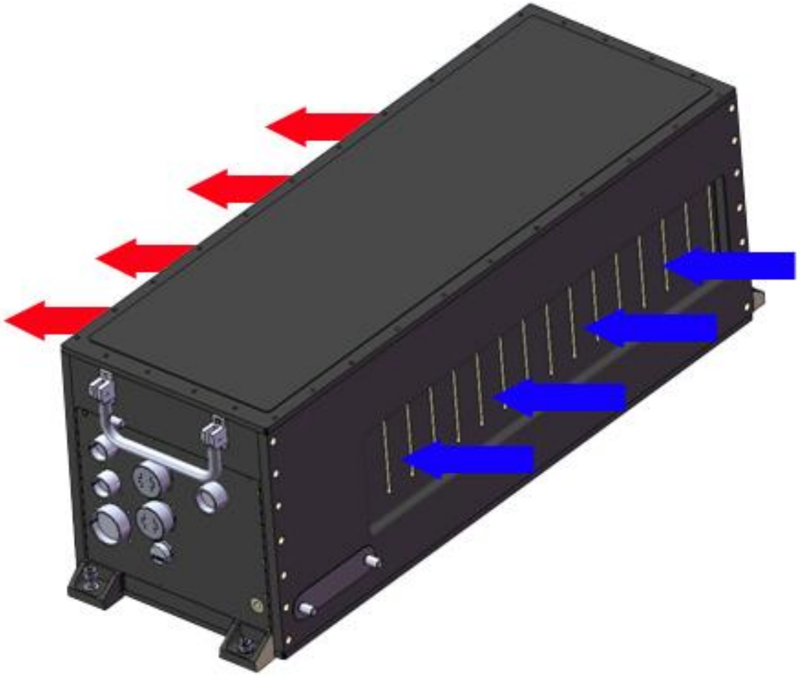


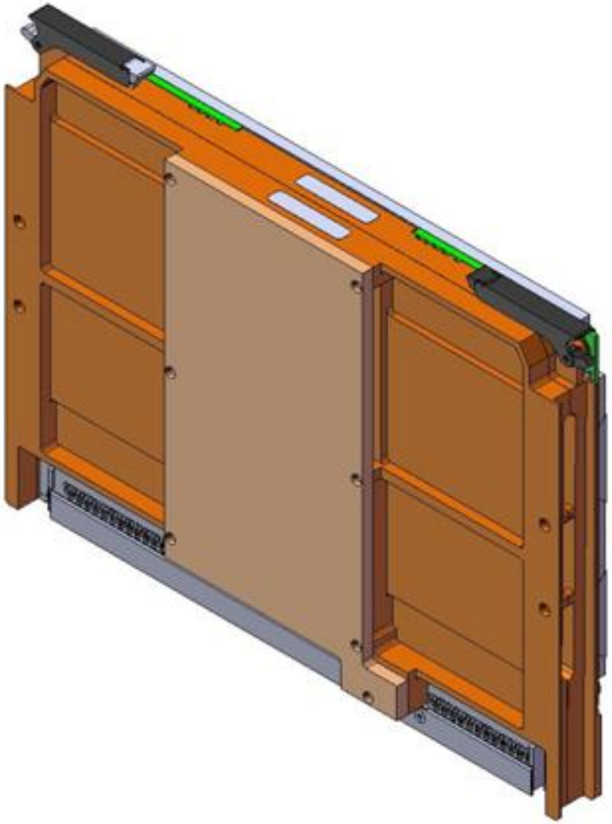
AFT Cooling - VITA 48.5

AFT Exploded View



AFT Module Features





Questions?

Seminar Handout

Tapping the TeraFLOP Potential of GP-GPU for High-Performance Computing Applications

Dr. Brooks Moses, Sourcerer, Mentor Graphics (brooks_moses@mentor.com)

Dr. Gil Ettinger, Consultant, Sensor Exploitation R&D (gil.ettinger@gmail.com)

Eran Strod, Curtiss-Wright Embedded Computing (eran.strod@curtisswright.com)

Resources:

NVIDIA CUDA developers site: http://developer.nvidia.com/category/zone/cuda-zone.

Mentor Embedded Sourcery VSIPL++: http://go.mentor.com/vsiplxx.

NVIDIA CUDA libraries: http://developer.nvidia.com/technologies/libraries.

CULAtools linear algebra library: http://www.culatools.com.

Richard Szeliski, *Computer Vision: Algorithms and Applications*, Springer, 2010 (http://szeliski.org/Book/, ISBN: 978-1-84882-934-3). A comprehensive reference on computer vision algorithms.

GPU implementations of low-level computer vision algorithms (University of Toronto): http://openvidia.sourceforge.net/index.php/OpenVIDIA.

Papers from IEEE Computer Vision & Pattern Recognition 2010 conference : http://www.cvpapers.com/cvpr2010.html.

Curtiss-Wright Embedded Computing: http://www.cwcembedded.com.

MXM graphics subsystem interface specification and standards body : http://www.mxm-sig.org.

Vita Standards Organization, VMEbus technology : http://www.vita.com.

PCI Express communications bus specification and standards body: http://www.pcisig.com.