SAND2012-3446 C
SAND2012-3446C

Lessons Learned Using NVidia GPUs Within SAR Applications

SAR/GPU Workshop

April 18-20, 2012

Donald Small
Embedded Radar Processing Department

Sandia National Laboratories

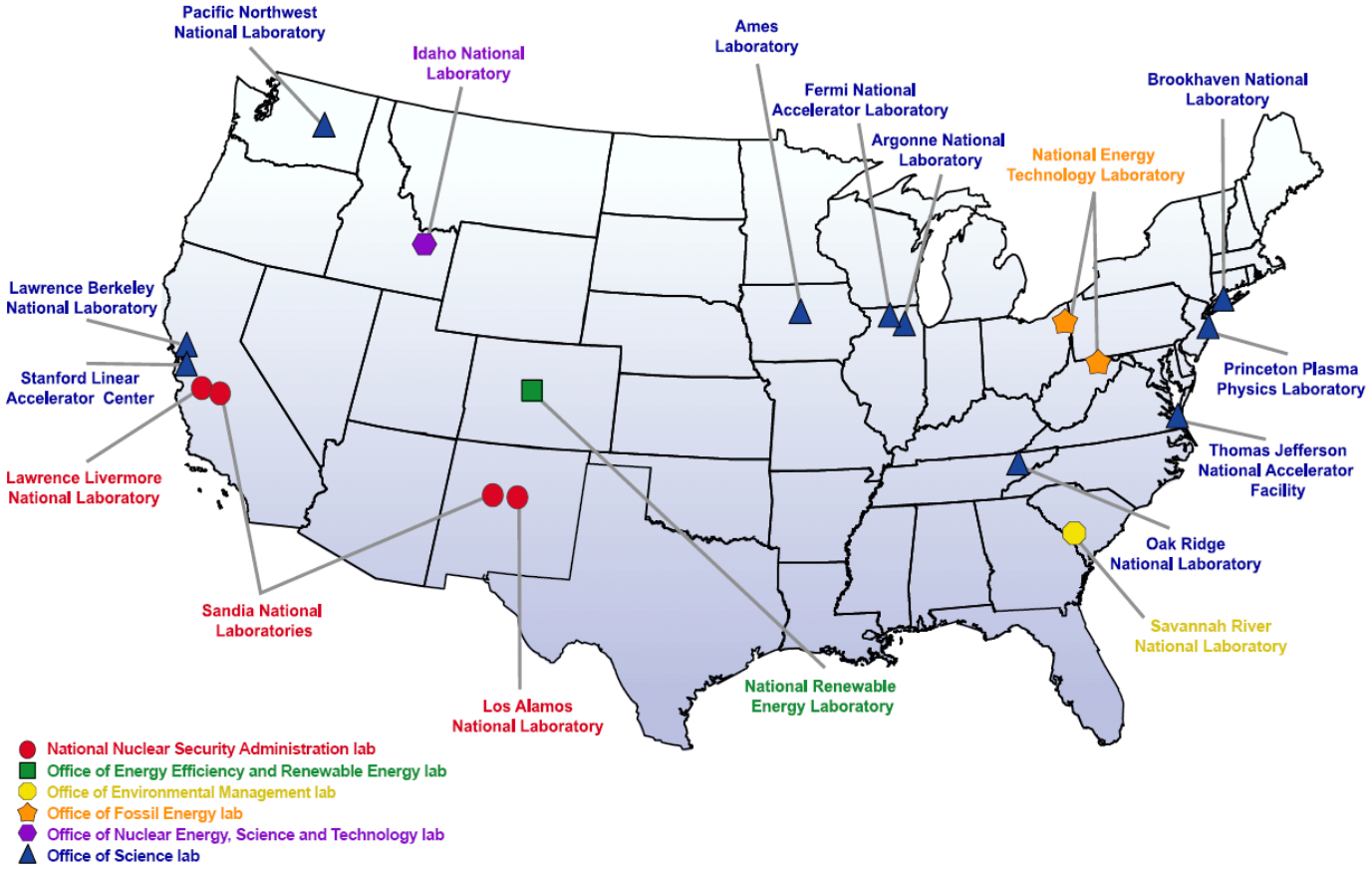
Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Sandia National Laboratories

Presentation Outline

- Introduction
- SAR Applications
- History of CUDA and Sandia SAR Processing
- Important Issues While Porting Your Application To CUDA
- Examples Within SAR Applications
  - Resampling
  - Correlation
  - Back Projection
  - Double precision
- Available HW
- CUDA

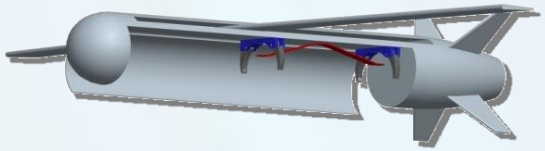
Department of Energy National Laboratories



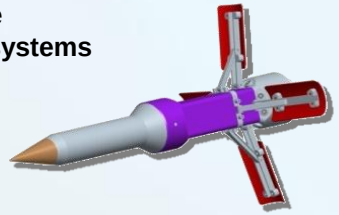
Sandia National Laboratories

Strategic National Security, Multi-Program Laboratory

Defense Systems & Assessments



Ground sensors for future combat systems



International, Homeland, and Nuclear Security

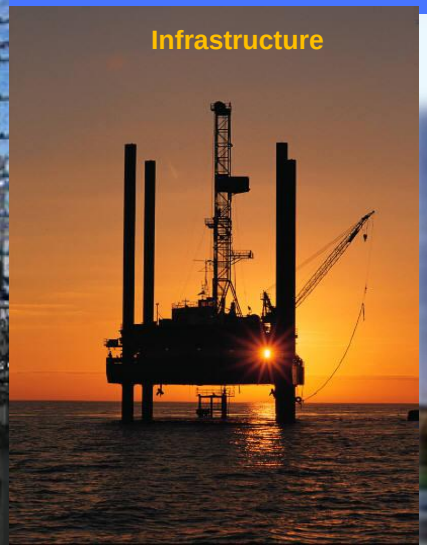
Critical Asset Protection

Homeland Security

Global Security

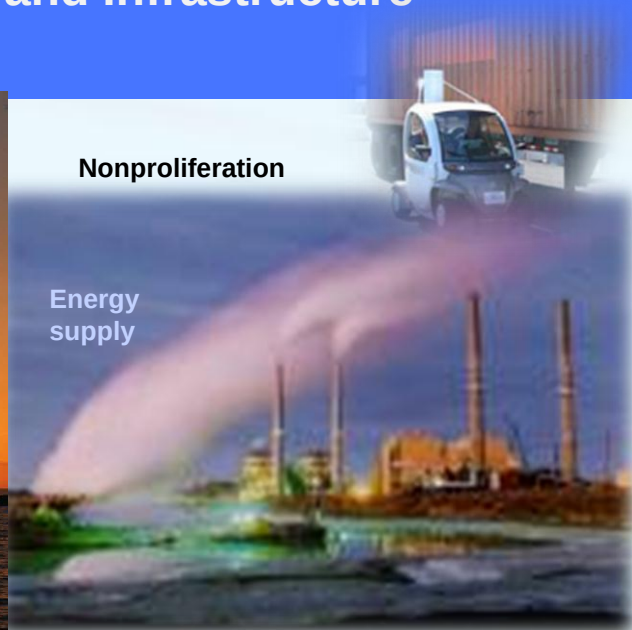
Homeland Defense & Force Protection

Energy, Climate, and Infrastructure Security



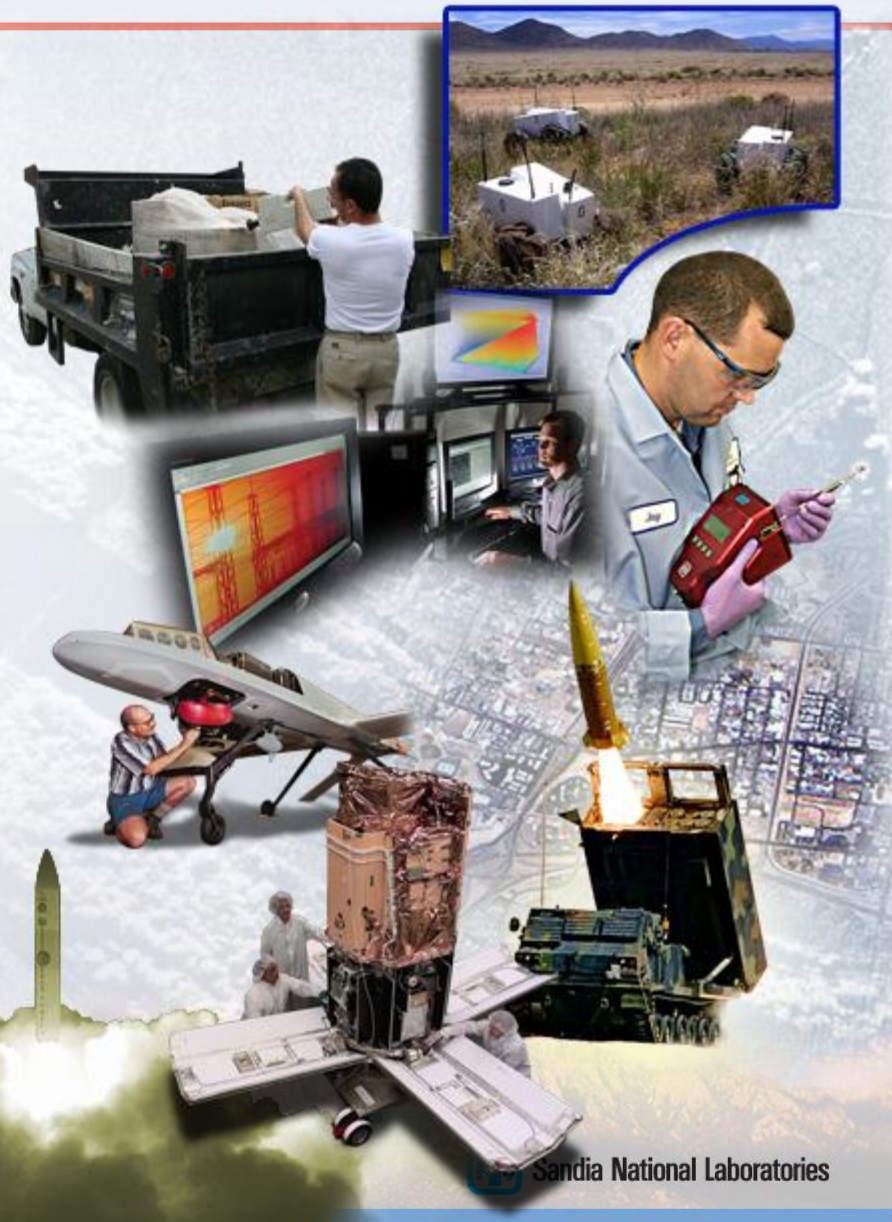
Nonproliferation

Energy supply



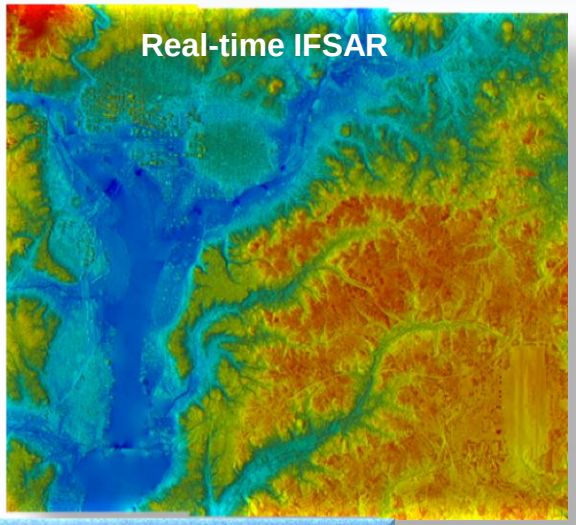
Defense Systems & Assessments Programs

- Science & Technology Products
- Surveillance & Reconnaissance
- Integrated Military Systems
- Remote Sensing and Verification
- Information Operations
- Space Missions
- Proliferation Assessment



Defense Systems & Assessments Programs

- Science & Technology Products
- **Surveillance & Reconnaissance**
- Integrated Military Systems
- Remote Sensing and Verification
- Information Operations
- Space Missions
- Proliferation Assessment



Program Thrusts:
- Military Space
- Sensor Systems
- Microwave ISR

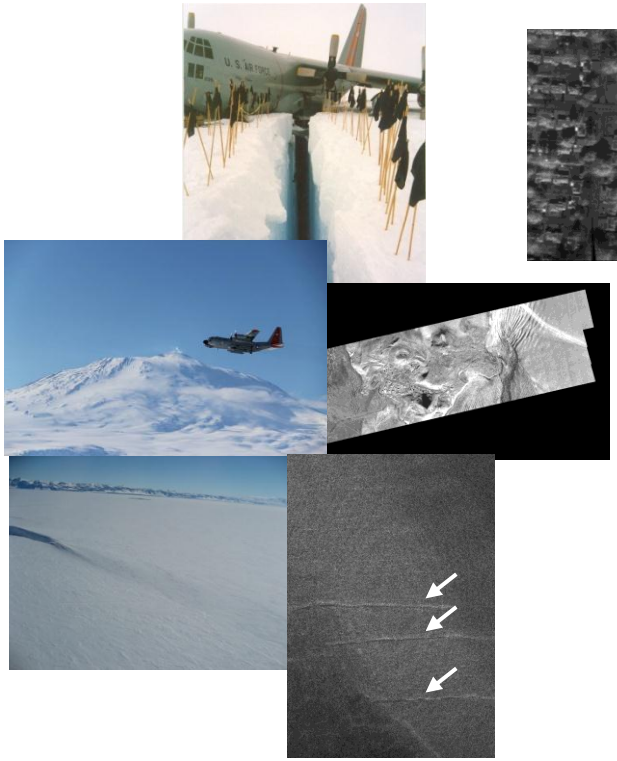
Highlights:
- Full modality, miniaturized radar systems
- Rapid Terrain Visualization
- Coherent Change Detection

Manned and Unmanned SAR

Sandia National Laboratories

Sandia Technology Engaged in a Wide Variety of Missions

Antarctica Crevasse Detection in support of NSF/NYANG (X-Band)

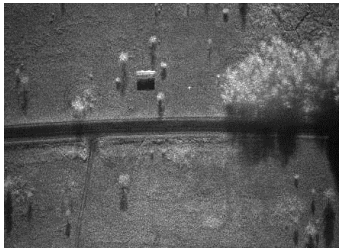


Crevasse Detection

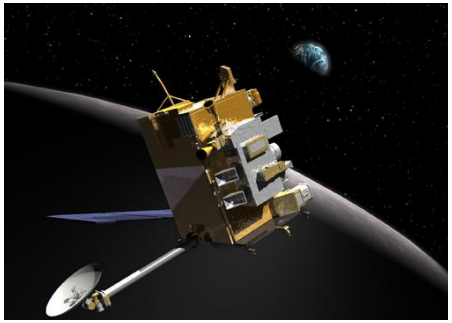
Multiple UAV or Manned Applications

Real-time, 0.1m resolution SAR on small UAVs

- Stripmap, spotlight, CCD images downlinked in real time to groundstation



Lunar Reconnaissance Orbiter Mission



Mini-RF Technology Demonstration (Sponsored by NASA/NAWC)

- Aided in location of subsurface water ice deposits. Imaged entire lunar surface, including high-resolution imagery of permanently-shadowed regions. (S-Band)
- Space-qualified version of MiniSAR core HW used in imaging system electronics

Real-Time Image

Tijeras Arroyo Golf Course: 4-inch resolution, 3.3 km range, 20050519:PASS005

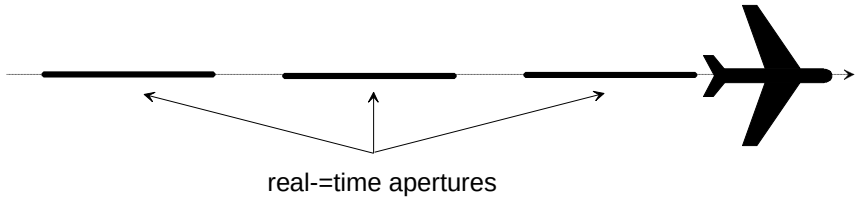
VideoSAR Mode

Sandia National Laboratories

Traditional SAR vs. VideoSAR

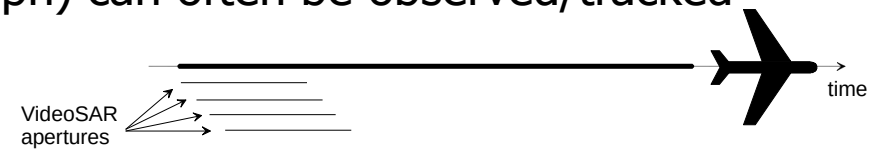
Traditional SAR

- phase histories are only collected during real-time apertures
- time between images = time to collect real-time aperture + time to process image (many tens of seconds at long ranges)
- Moving targets disappear or smear, difficult to locate/track



VideoSAR

- phase histories are collected continuously
- images are formed from overlapping sets of phase histories
- time between images is user selectable and is independent of aperture length (0.1 to 0.3 seconds seems best)
- slow moving targets (< 15 mph) can often be observed/tracked
- Latency < 8 sec.



A rapid sequence of SAR images (> 1 Hz) can permit observation of target shadows

Real-Time Image

Eubank Gate, KAFB: 4-inch resolution, 3.3 km range, 20050519:PASS007

Video SAR
(Movie Clip)

History of CUDA and Sandia SAR Processing
Polar Format

- **Start January 2007.**
  - Two man months for conversion of major portions of algorithm.
  - Proof of concept for new development program.
  - Three man months for all functions.
    - Includes range/azimuth window application, range and azimuth compression, multiple corrections, azimuth interpolations and phase gradient autofocus.
  - Performance
    - 2k x 2k image
    - 1.8GHz Intel Zeon — 4.5 Seconds
    - 8800 GTX (128 cores) — 150 mS
    - Speedup 30X

Sandia National Laboratories

History of CUDA and Sandia SAR Processing
Overlap Subaperture Polar Format, VideoSar

- **Overlap Subaperature Polar Format – Designed for embedded multiprocessors with distributed memory systems. Includes many additional corrections such as antenna pattern correction, digital receiver filter corrections, range curvature corrections, etc.**
  - Started September 2007

- **VideoSar – design requirement 1k x 1k image @ 5 FPS, goal of 1k x 1k image @ 10 FPS.**
  - C1060 (240 cores) – 1k x 1k image @ 14 FPS
  - C1060 (240 cores) – 2k x 2k image @ 3 FPS

History of CUDA and Sandia SAR Processing
Coherent Change Detection

- **6 man months to convert to GPU.**
  - Much of which was parallelizing the tie point correlation function.
  - Performance
    - 3k x 5k pair of images.
    - Quadro 5010M (348 cores) 700 mS

History of CUDA and Sandia SAR Processing Back Projection

- **Developed directly for GPU. No autofocus at this time. Digital Elevation Map input, non rotating coordinate frame.**
- **Performance 2k x 2k image**
  - Quadro FX 3600 (128 cores) 4 Seconds
- **VideoSar Backprojection**
  - Quadro FX 3600 (128 cores) 250mS or 4 FPS

Important Issues While Porting To CUDA Memory Bandwidth Host to Device

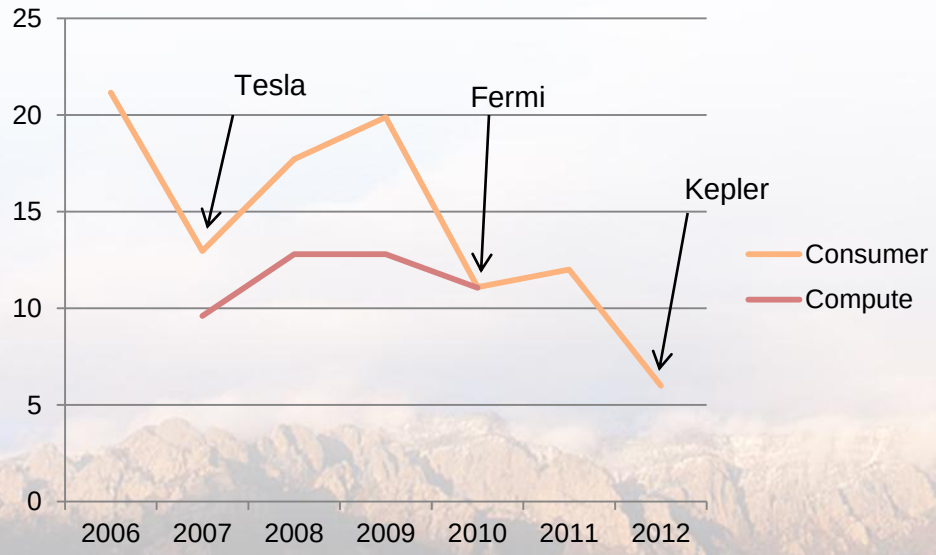
- **Most if not all Nvidia GPUs are PCIe x16 (16 bidirectional lanes) of data.**

PCIe version	Theoretical	Measured
1.0	4 GB/s	2.5 - 3.5 GB/s
2.0	8 GB/s	5.0 – 7 GB/s
3.0	16 GB/s	Unknown

- **GPU compute capability 1.0 had ability to transfer data in a single direction at a time**
- **GPU compute capability 2.0 has ability to transfer data bidirectional (as well as run a kernel) concurrently**

Important Issues While Porting To CUDA
Memory Bandwidth Host to Device

- **The best case theoretical data transfer rate from host to GPU is 32GB/s (PCIe v3.0), only 1/6 the theoretical data rate of best case global memory to cores (192 GB/s)!**
  - PCIe v2.0 has had a run of almost 5 years starting at 1/11 (GTX 8800) and performing as poorly as 1/20 (GTX 285) before Fermi class GPUs gave us bidirectional data transfer which only reset the ratio.
  - And this ratio will only get worse until PCIe v4.0 is accepted.



Sandia National Laboratories

Important Issues While Porting To CUDA Memory Bandwidth Host to Device

- **What does this mean?**
  - Perform serial processes on CPU, parallel processes on GPU!
  - Reduce Host->Device or Device->Host transfers
  - Don’t transfer small chunks of data, combine into largest possible structure for data transfer.
  - Use bidirectional transfer of data and concurrent kernel invocation.
  - Once the data is on the GPU, keep it there.
    - Example: Phase Gradient Autofocus (PGA)
    - Much of PGA is parallel, until all of the phase errors are collapsed into a single vector.
    - Ratio of CPU vs GPU processing time ~ 1.0

Important Issues While Porting To CUDA Memory Bandwidth Global Memory

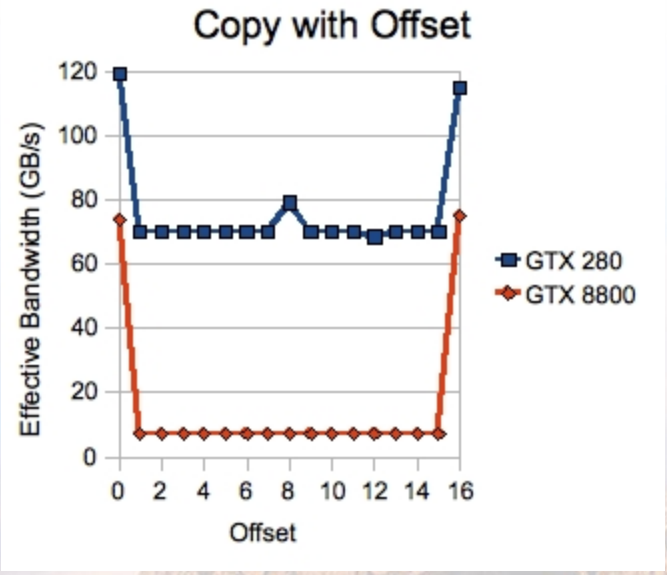
- **What is it?**
  - Large bulk storage for data. 768 MB (8800 GTX) to 6 GB (M2070, M2090)
  - Fast – originally 84.6 GB/s now reaching 192 GB/s
  - Latency from kernel request to register is 600 – 800 cycles.
    - If blocks are created with enough free resources (see Nvidia occupancy calculator), GPU H/W will context switch up to 1536 threads per multiprocessor to hide latency
  - Accessible from all threads at all times.
    - Must be careful about multiple threads writing to common locations. GPU does not guarantee that thread 0 runs before thread N outside of a WARP.
  - See section 2.2 about “Bandwidth” in the “CUDA C Best Practices Guide” to learn more about theoretical and effective bandwidth calculations.

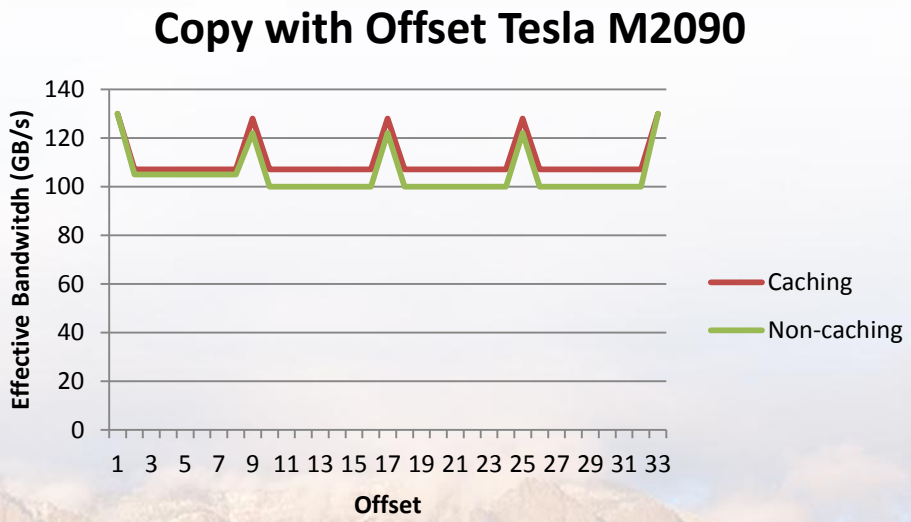
Important Issues While Porting To CUDA
Memory Bandwidth Shared Memory

- Small storage area
- Very Fast – 1 clock cycle, if you follow all of the rules!
- Accessible from all threads within a thread block.
- Not persistent after thread block completes.
- Great for small data arrays of common use to all threads within a thread block.
- May be treated as a user managed high speed cache.
- Example
  - A range dependent phase term needs to be applied to all phase history data. The phase term may be calculated by one thread, saved to shared memory and finally read and applied by all threads.

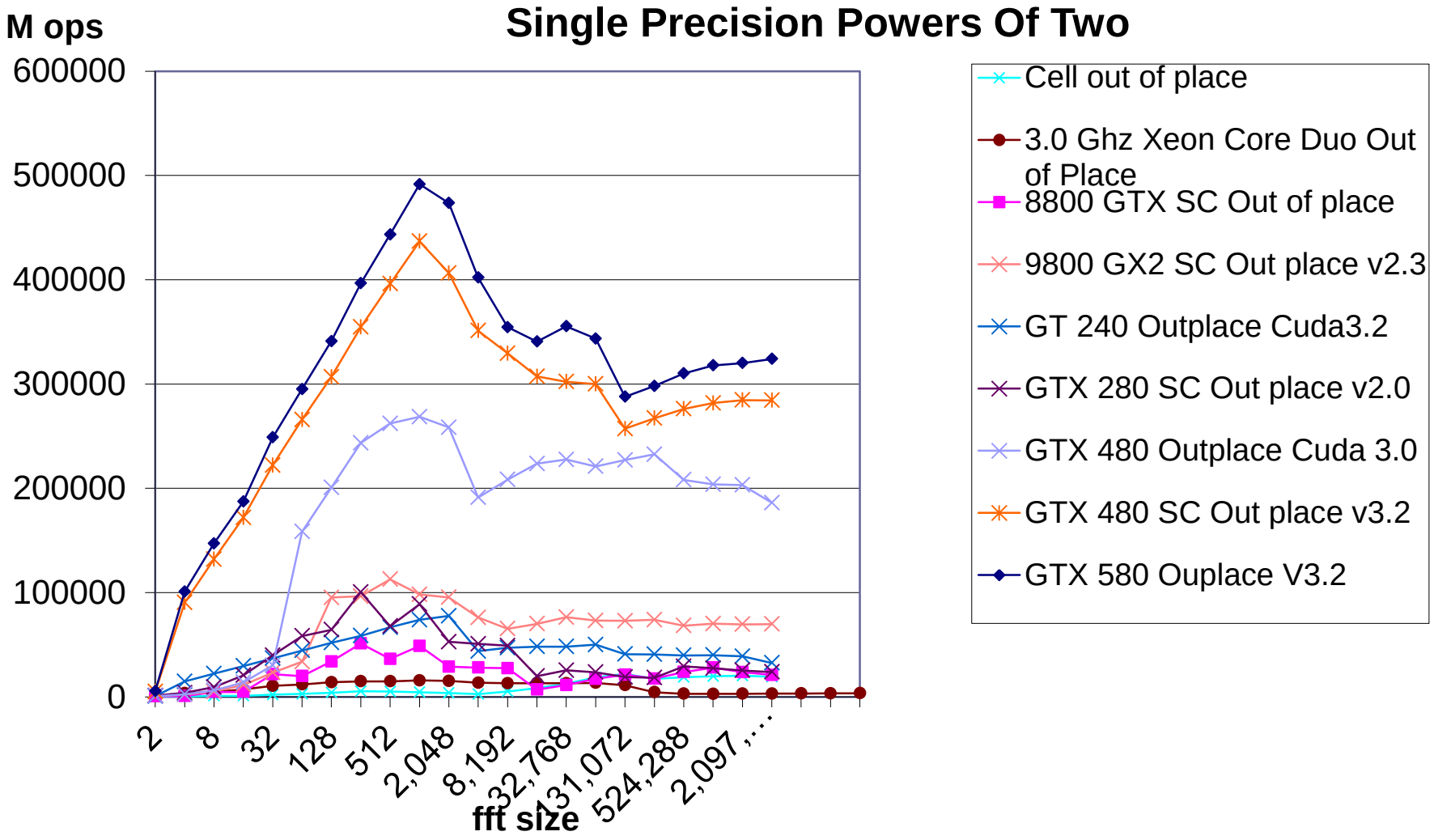
Memory Access Tips

- **Coalesce access to global memory.**
  - See section 3.2.1 about “Coalesced Access to Global Memory” of the “CUDA C Best Practices Guide” for more information.
  - Rules for coalescing memory accesses are dependent on compute capability
  - Memory access alignment is critical for maximum performance

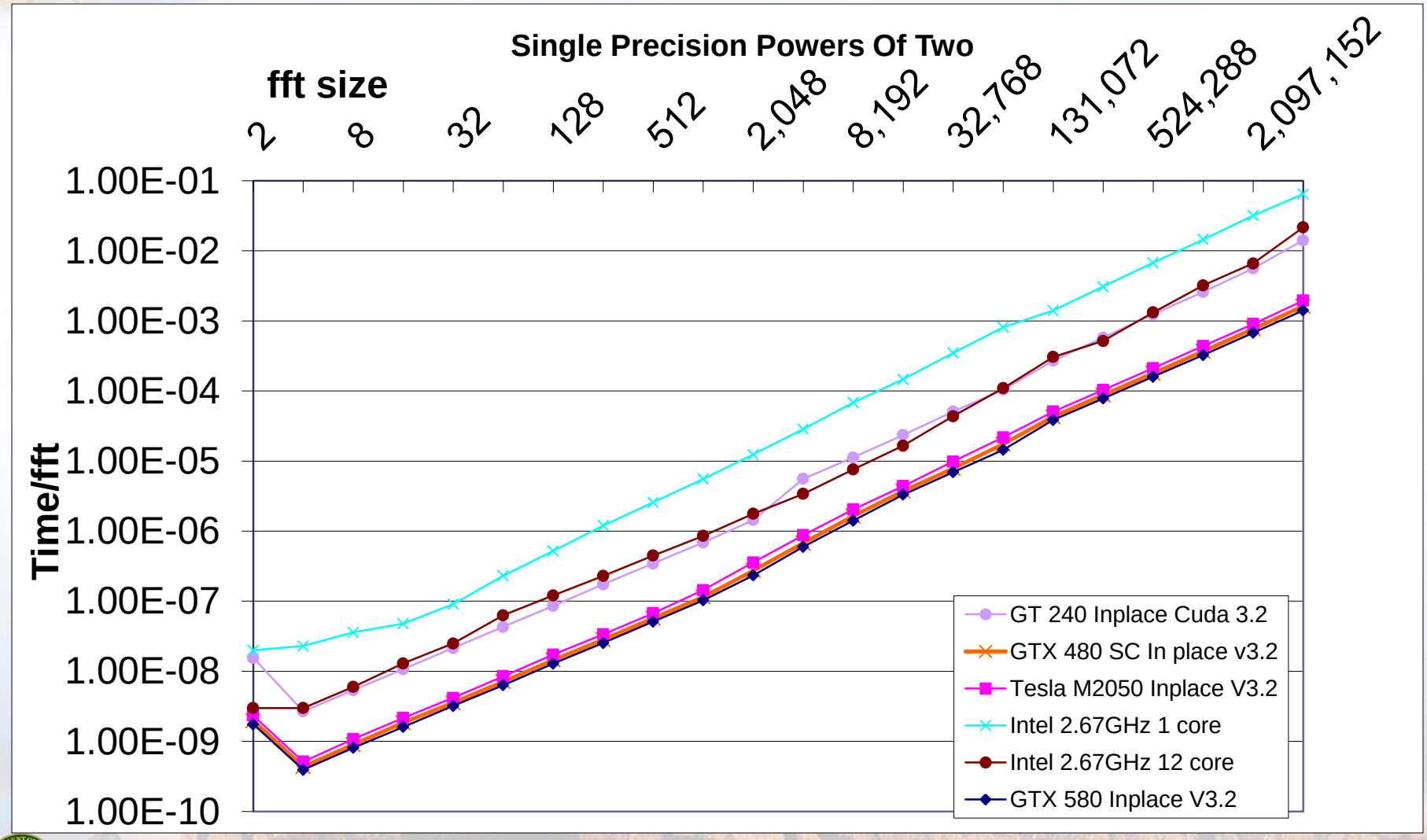




Computational Bandwidth – FFT Benchmarks



Computational Bandwidth – FFT Benchmarks



Computational Bandwidth
I/O Versus Computation Bound

- **Consider a CUDA Magnitude**

```
__global__ void  Mag1DKernel(const float2 *in, float *out)
{
   //  Determine my thread ID
   const unsigned int myX = (blockIdx.x * blockDim.x) + threadIdx.x;

   //  Fetch one complex sample from global memory
   float2 data = in [ myX ];

   //  Calculate the magnitude of the complex value and store
   out [ myX  ] = sqrtf((data.x * data.x) + (data.y * data.y));
}
```

- **Is this kernel compute or IO bound?**
  - Assume a GPU with 192GB/s global memory bandwidth.
  - Assume kernel attains the maximum bandwidth
    - accessing memory in a coalesced fashion
    - data properly aligned.

Computational Bandwidth
I/O Versus Computation Bound

- **Is this kernel compute or IO bound?**
  - Depends on the number of threads launched to make sure we cover global memory read/write latency.
- **Now assume launch with 512 threads. Is this kernel compute or IO bound?**
- **How to determine.**
  - Run a test program with multiple launches of this kernel and one of the several performance measurement tools that NVidia provides and get a measure of average performance.
  - Modify the kernel and add extra computation burden.
    - Be careful, the compiler is very smart about common sub expression reductions, unnecessary memory accesses, etc.
  - Rerun and measure average performance.

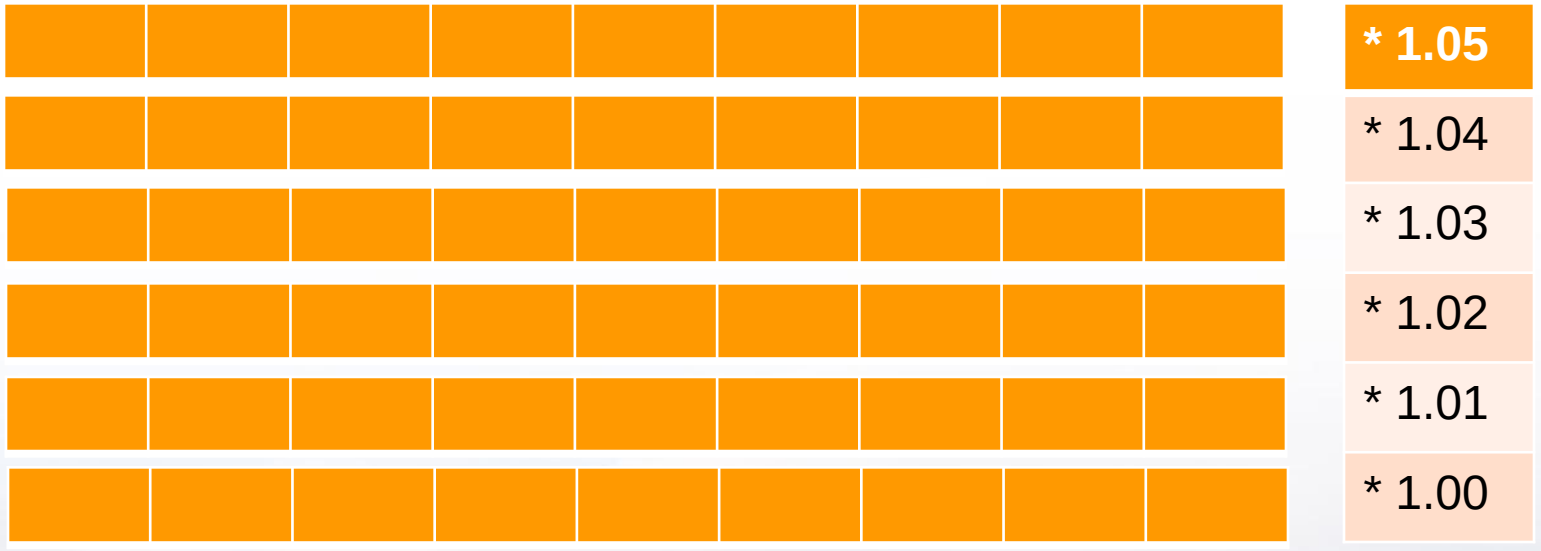
Computational Bandwidth
I/O Versus Computation Bound

- **Observation – added 25 additional operations before the run time increased.**
  - Caveat: heavily memory alignment and hardware (compute capability) dependent

- **Discovery – We estimate that 60% of our kernels are I/O bound.**
  - Examples of not complex enough to become compute bound:
    - Applying a phase correction
    - window functions,
    - antenna amplitude corrections
  - Any extra computations that can be added until the kernel becomes compute bound are FREE!

Computational Bandwidth
Think Big

- Consider applying a range dependent antenna amplitude correction to an image. A scalar vector multiply.



- cuBlas has a scalar vector multiply function.

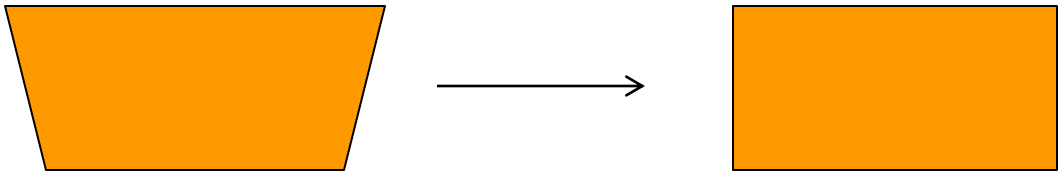
Computational Bandwidth
Think Big

- **Lesson Learned: Launch 1 large user defined function versus many cuBlas function calls.**
  - Hardware supports 4 or more concurrent kernel launches.
  - Today's CUDA provides asynchronous kernel launches with streams, but the kernel launch still has some small overhead.
  - Assume 2-5uS per kernel launch and for an image with 4000 range lines = 8-20mS of wasted time.

- **Lesson Learned: Launch kernels with as many threads as possible.**
  - 5 years ago, the 8800 GTX had 128 cores. 5 years later, we have 1536 cores available.
  - GTX 680 can have over 20,000 concurrent threads with HW context switching.
  - Take advantage of conditional compilation or cudaGetDeviceProperties() to scale over time.

NVidia Accelerator Technology - Resampling

- **Azimuth resampling**



  - **Resample techniques**
    - Chirp Z
    - Sinc Interpolation
    - Others
  - **Order of preference for speed: CPU**
    - Chirp Z, Sinc (hand coded SSE implementation), Others
  - **Order of preference for speed: GPU (2k x 2k)**
    - Sinc 8mS, Chirp Z 50mS, Others

- **Lesson Learned – just because a method is the fastest on the CPU does not mean it will be fastest on NVidia GPUs**

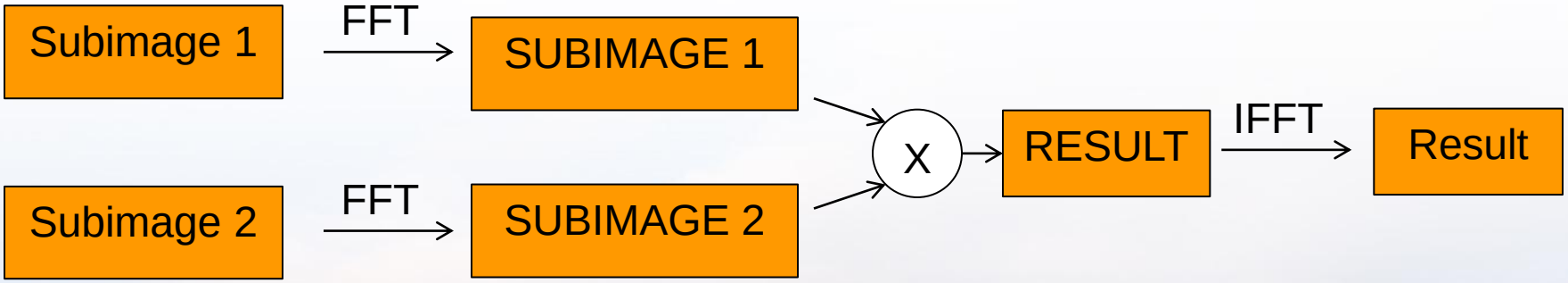
NVidia Accelerator Technology – Resampling

- **Methods of performing Sinc interpolations (CPU)**
  - Calculate sin(x)/x for each zero crossing – slow, most precise
  - Table driven- fastest, precision good enough
    - lookup nearest neighbor
    - Linear interpolation between table points
- **Methods of performing Sinc interpolations (GPU)**
  - Table driven- slow, precision good enough
    - lookup nearest neighbor
    - Linear interpolation between table points
  - Calculate sin(x)/x for each zero crossing – fast, most precise
  - Calculate sin(x)/x with fast math – fastest, better than table
- **Lesson Learned – parallel transcendental functions are faster than table lookup (even when using templates, which are cached)**

NVidia Accelerator Technology - Correlation

- **Correlation**
  - CPU serial pseudo code

```
foreach grid location in x
        foreach grid location in y
                Correlation(image#1(x, y, boxsize), image#2(x, y, boxsize))
        next y
next x
```

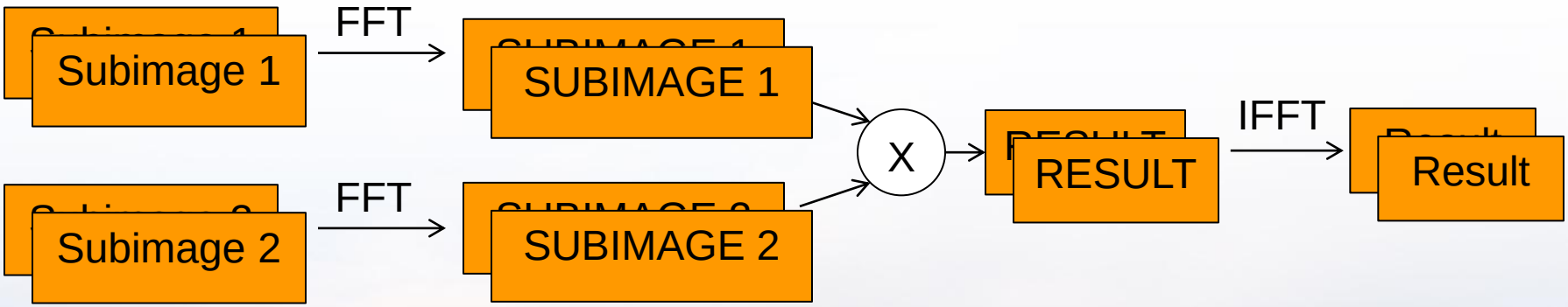


- **GPU underutilized**
  - Small correlation areas of 64 x 64 pixels or less

NVidia Accelerator Technology - Correlation

- **Pseudo Code**

```
foreach grid location in x
        foreach grid location in y
                listOfGridPoints += x, y
        next y
next x
Correlation(image#1(listOfGridPoints, boxsize), image#2(listOfGridPoints , boxsize))
```



- **Lesson Learned: Increase compute density by increasing parallelism.**
  - In this case, at the expense of much more memory!

NVidia Accelerator Technology – Back Projection

- **Resampling Choices Effect Memory Access Patterns**
  - Use of zero padded FFTs. Nearest neighbor data access. Threads may not read consecutive memory, ie coalesced memory locations.
    - Use cached memory, textures, L1 cache, etc
  - Interpolation filters such as windowed sinc, etc. Highly overlapped memory access patterns.
    - Use shared memory
- **Textures**
  - See section 5.3.2.5 “Texture and Surface Memory” in the “CUDA C Programming Guide” for additional information.
  - Optimized for 2D spatial locality.
  - Cached, but note from “C Programming Guide”, a cache hit reduces DRAM bandwidth demand but not fetch latency.
  - Can not access double precision data types!
  - Maximum dimension sizes for 1D textures may cause limitations.

NVidia Accelerator Technology – Back Projection

- **Double Precision**
  - Polar format, Overlapped Subaperture, Coherent Change Detection. Data path analysis has shown single precision works well. Precision issues can arise if sums of data are necessary (GMTI).
  - Back Projection. Double Precision is necessary for range calculations. Your choice for all other calculations.
  - See Portillo, R. “Power versus Performance Tradeoffs of GPU-accelerated Backprojection-based Synthetic Aperture Radar Image Formation”, SPIE 2011

Nvidia Hardware Lines

- **Consumer and Mobile - GeForce**
  - Cheap, available at local computer stores
  - If it fails, upgrade to faster HW
  - Perfect for learning CUDA and creating/debugging many kernels
  - Little to no double precision support
  - NVidia does not guarantee correctness of calculations
- **Commercial - Quado Line**
  - Expensive! Nvidia guarantees replacements available for 3 yrs
  - Double precision support available (dependent on version)
  - NVidia does not guarantee correctness of calculations
- **Scientific - Tesla Line**
  - Expensive! Nvidia guarantees replacements available for 3 yrs
  - Double precision support available (dependent on version)
  - NVidia guarantees correctness of calculations – clocked back to insure
  - 6 GB global memory available

Nvidia Hardware Lines

- **Embedded products are available**
  - Curtiss-Wright - 1 generation of products
  - GE - 2 generations of products

CUDA

- **Register as a NVidia developer for early access to CUDA releases and developer support.**
- **CUDA has been very stable over its lifetime!**
  - I have CUDA Beta Prerelease 0.2a – 4.2
  - We have found one issue in 5 years of use. NVidia promptly solved the problem.
- **CUDA is updated on a 6 month schedule.**
- **Very proactive on adding new capabilities.**