14 July 2010

Using GPU VSIPL & CUDA to Accelerate RF Clutter Simulation

2010 Symposium on Application Accelerators in High Performance Computing

Dan Campbell, Mark McCans, Mike Davis, Mike Brinkmann
dan.campbell@gtri.gatech.edu

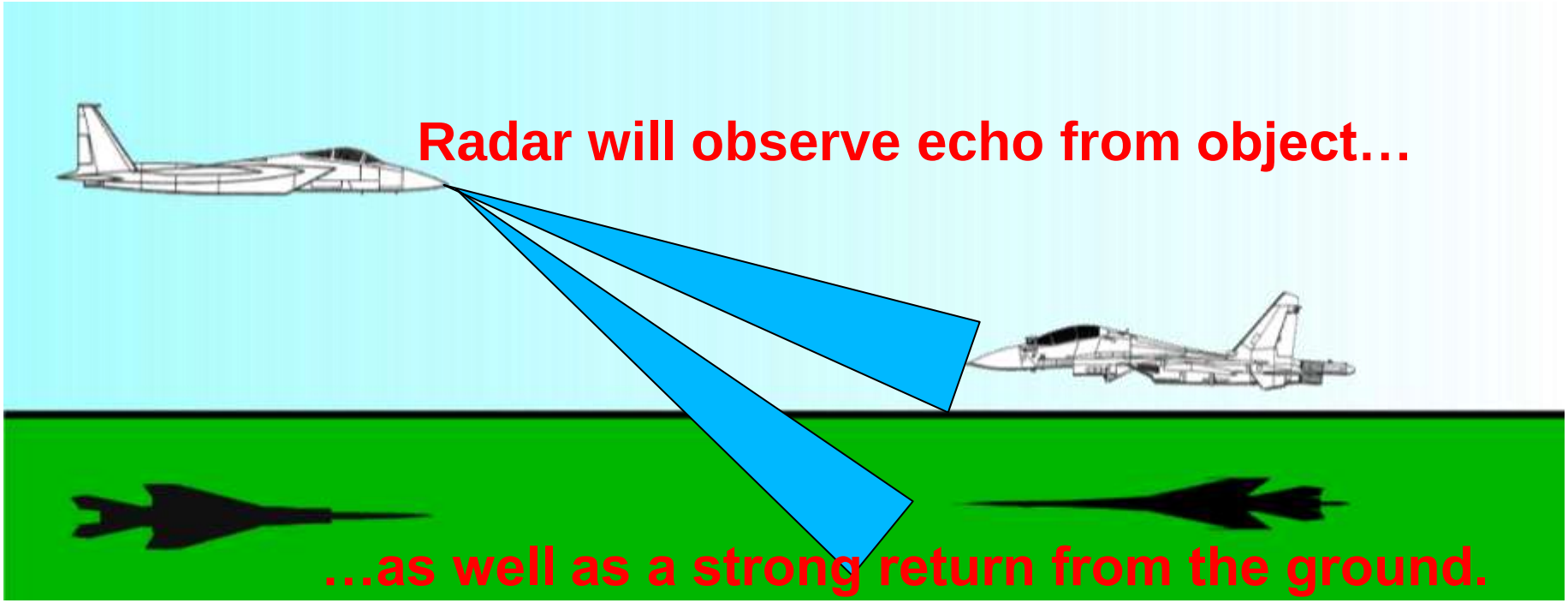
Outline

- RF Clutter Simulation
- Validation Approach
- GPU VSIPL
- Precision Issues
- VSIPL Port, Optimization, and Results

Outline

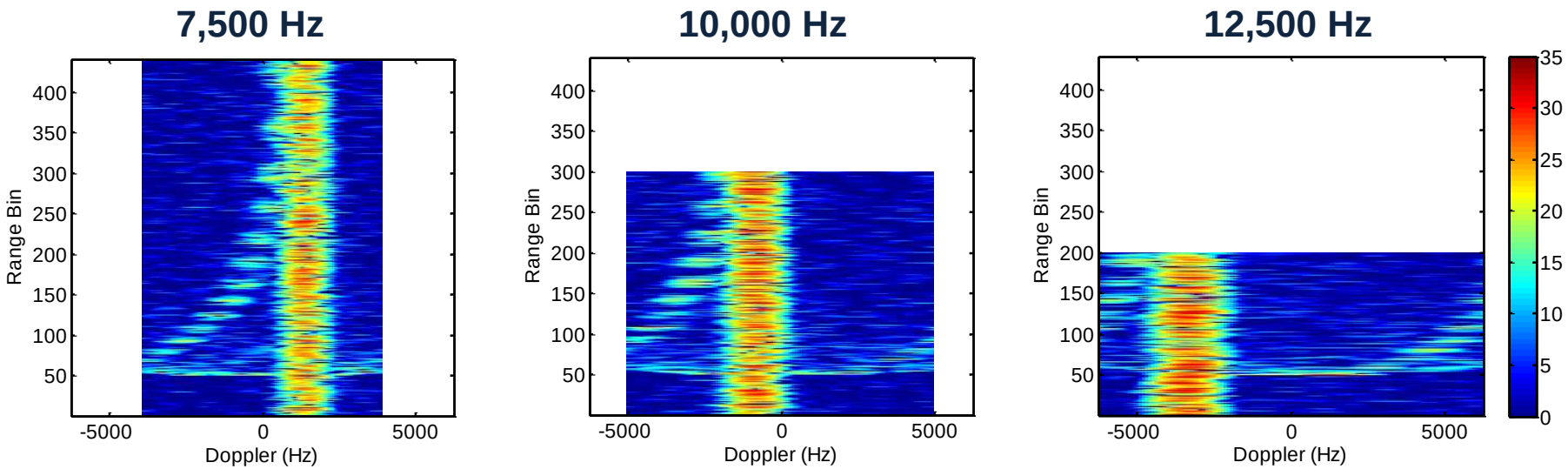
- RF Clutter Simulation
- Validation Approach
- GPU VSIPL
- Precision Issues
- VSIPL Port, Optimization, and Results

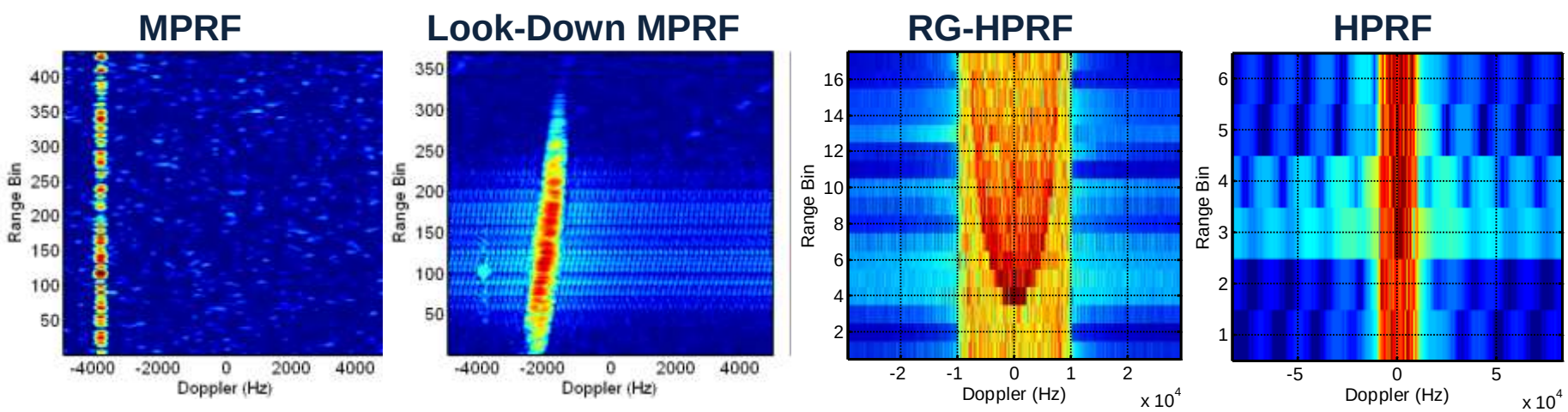
Radar Clutter



Strong returns from the ground, called "clutter", often limit the performance of radars in air-to-air and air-to-ground operations.

Synthetic Air-to-Air Clutter

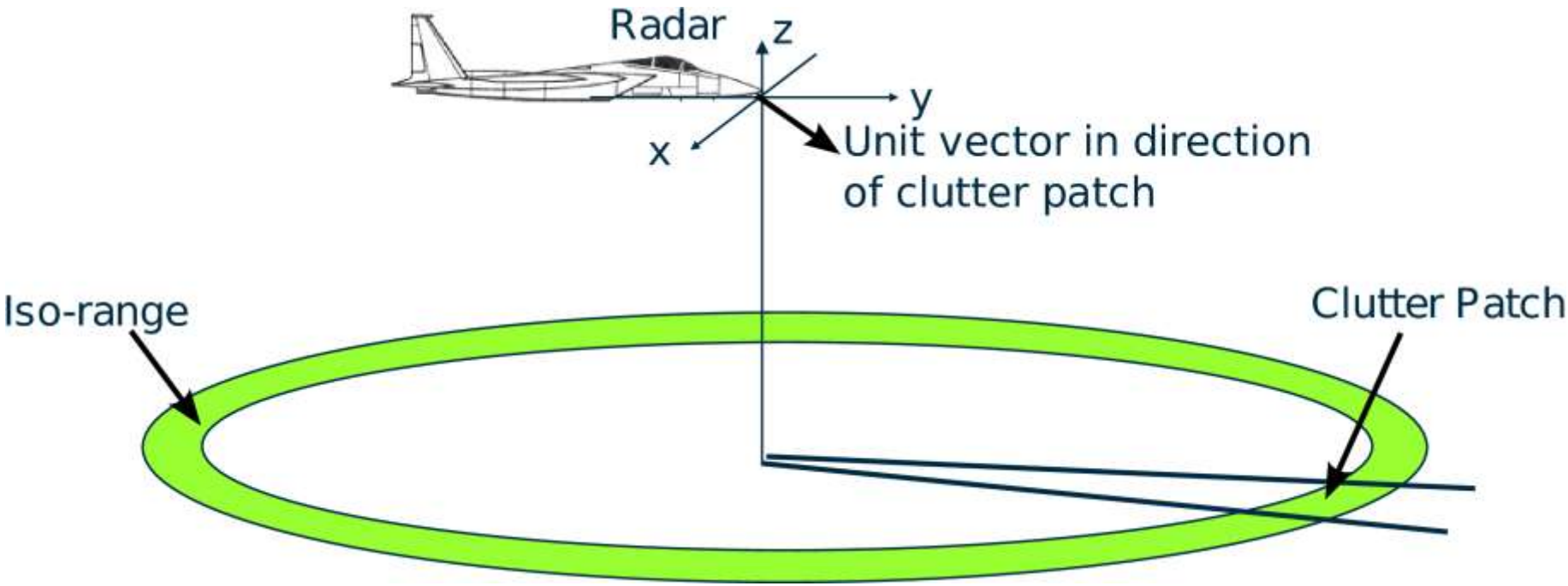




Targets at same range/Doppler as clutter will be obscured.

RF Clutter Simulation

<u>Approach</u>: Sub-divide ground into number of unresolvable clutter patches and compute contribution of each.



RF Clutter Simulation

Notional Parameters

	Air-to-Air	SAR Imaging (Air-to-Ground)	Our Test
# of Range Bins	200	1750	500
# of Pulses	128	3000	8
# of Clutter Patches	6,800 Rng x 96 Az = 6.5×10^5	14,500 Rng x 26,812 Az = 3.8×10^8	566 rng x 52 az = 29,432

Computational load depends on radar parameters and collection geometry (e.g., high resolution scenarios require a large number of independent clutter patches)

RF Clutter Simulation

Algorithm:

Inputs

- Radar Parameters (waveform, antenna, etc.)
- Location of platform for each pulse

Output

- Simulated radar data cube (sample voltage for each pulse, each channel, and each range bin)

For each pulse and for each range bin…

For each clutter patch in this range ring…

1. Compute range, azimuth, and elevation from platform to clutter patch.
2. Scale contribution of this clutter patch according to the radar range equation.
3. Accumulate the contribution of this clutter patch to the simulated data cube.

Outline

- RF Clutter Simulation
- Validation Approach
- GPU VSIPL
- Precision Issues
- VSIPL Port, Optimization, and Results

Validation Needs

- **Porting MATLAB ➔ C introduces changes**
  - Random Number Generator
  - Double ➔ Single
  - Implementation of some functions *e.g.* transcendentals
  - Reordering of operations
  - Programmer Error
- **Identical output too costly**
- **Derive acceptance criteria from expected usage needs**

Validation Approach

- Modify sim to capture RNG stream from MATLAB
- Automate large number of runs for golden data
- Accelerated port optionally ingests RNG stream
- Capture port output and compare to golden data
- Acceptance Criteria:
  - $CNR\Delta = (CNR_M - CNR_T) / CNR_M < 10^{-4}$
  - ECR = 20 log10(norm(M(:) - T(:)) / norm(M(:))) < -60dB
  - $ADMSE = Mean(| fft2(M(:)) - fft2(T(:)) |^2) < 10^{-3}$

Outline

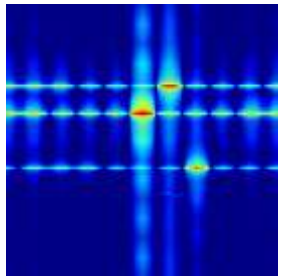
- RF Clutter Simulation
- Validation Approach
- GPU VSIPL
- Precision Issues
- VSIPL Port, Optimization, and Results

VSIPL

Vector Signal Image Processing Library

The Open Industry Standard For Signal Processing

- http://www.vsipl.org
- Industry standard C API for *portable* dense linear algebra & signal processing
  - Also C++, Python
- Accelerated implementations for many platforms, primarily embedded, coprocessor-based systems
- GPU VSIPL VSIPL implementation that exploits Graphics Processing Units to accelerate VSIPL applications – developed at GTRI
  - http://gpu-vsipl.gtri.gatech.edu

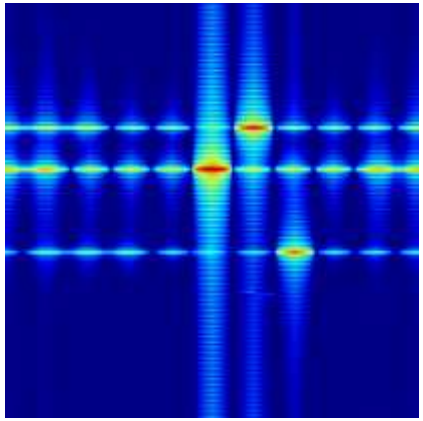


Simple VSIPL Application

```
#define ARRAY_SIZE 50

int main (int argc, char *argv[])
{
        vsip_vview_f  *v1, *v2, *v3;
        int i;

        v1 = vsip_vcreate_f (ARRAY_SIZE, VSIP_MEM_NONE);
        v2 = vsip_vcreate_f (ARRAY_SIZE, VSIP_MEM_NONE);
        v3 = vsip_vcreate_f (ARRAY_SIZE, VSIP_MEM_NONE);

        vsip_vramp_f (0, 0.1, v1);
        vsip_vramp_f (0, 1.0f/15.0f, v2);
        vsip_vramp_f (0, 0.0, v3);

        vsip_vadd_f (v1, v2, v3);

        for (i=0; i<ARRAY_SIZE; i++)
        {
                printf ("out[%i]:%f\n", i, vsip_vget_f(v3, i));
        }
}
```

Application Example: Range Doppler Map

- **Simple Range/Doppler data visualization application demo**
- **Intro app for <u>new VSIPL programmer</u>**
- **431x Speedup TASP ➔ GPU-VSIPL**
- **<u>No changes to source code</u>**



Section	GT200 Time (ms)	Q6600 Time (ms)	Speedup
Admit	0.74	0.0	0
Baseband	1.46	1872.3	1280
Zeropad	1.91	110.7	58
Fast time FFT	1.38	5696.3	4130
Multiply	13.5	33.9	2.51
Fast Time FFT^{-1}	1.44	5729.0	3980
Slow time FFT, 2x CT	9.41	3387.0	360
log10 $\lvert . \rvert^2$	1.41	470.2	333
Release	9.44	0.0	0
Total:	40.7	17299.42	431

Outline

- RF Clutter Simulation
- Validation Approach
- GPU VSIPL
- Precision Issues
- VSIPL Port, Optimization, and Results

Original Validation Results

- VSIPL versions compared to MATLAB version

	VSIPL Double	VSIPL Single	Threshold
CNR Consistent	Yes	Yes	
CNR Δ	10^{-16}	10^{-6}	$< 10^{-4}$
ECR	-152 dB	2.9 dB	< -60 dB
ADMSE	10^{-12}	10^{4}	$< 10^{-3}$

Single Precision

- **Single precision errors caused by high dynamic range in platform to clutter patch range calculation:**
  - d(Platform➔clutter) **>>>** d(clutter patch➔clutter patch)
- **Solution: use far-field approximation technique**
  - Double precision used to compute a base range
  - Single precision for sets of ΔR values
  - Small number of double precision calculations has negligible affect on performance

Far Field Approx. via Taylor Expansion

Range between platform at x and clutter patch at y

$$R(\mathbf{x}) = \|\mathbf{x} - \mathbf{y}\|$$

Linear approximation near x_0

$$R(\mathbf{x}) \approx R(\mathbf{x}_0) + \underbrace{\overbrace{\left(\frac{\mathbf{x}_0 - \mathbf{y}}{\|\mathbf{x}_0 - \mathbf{y}\|}\right)}^{\text{Unit vector from CPI center to clutter patch}} \cdot \overbrace{(\mathbf{x} - \mathbf{x}_0)}^{\text{Distance from center of scene, } \epsilon}}_{\text{Distance travelled in direction orthogonal to "lines" of constant range}}$$

Quadratic Term

$$\frac{1}{2}\left[\frac{\|\epsilon\|^2}{\|\mathbf{x}_0 - \mathbf{y}\|} - \left(\epsilon \cdot \frac{\mathbf{x}_0 - \mathbf{y}}{\|\mathbf{x}_0 - \mathbf{y}\|}\right)^2\right] \approx 0 \text{ for } \|\epsilon\| << \|\mathbf{x}_0 - \mathbf{y}\|$$

Bounding Error

Approximation Error

$$\left| R\left(\mathbf{x_0} + \epsilon\right) - \hat{R}\left(\mathbf{x_0} + \epsilon\right) \right| \leq \frac{1}{2} \frac{\|\epsilon\|^2}{\|\mathbf{x}_0 - \mathbf{y}\|}$$

Case 1: Air-to-Air

128 pulses, 20 kHz PRF, 300 m/s velocity → $\|\epsilon\| < 1\mathrm{m}$

10 km Altitude → $\|\mathbf{x}_0 - \mathbf{y}\| < 10\mathrm{km}$

error < 50 μm < 0.06° phase at X band

~~Case 2: SAR~~

10 second dwell, 100 m/s velocity → $\|\epsilon\| < 500\mathrm{m}$

10 km Altitude → $\|\mathbf{x}_0 - \mathbf{y}\| < 10\mathrm{km}$

error < 12.5 m >> λ at X band!!!

Linear approximation to range may be appropriate for typical air-to-air scenarios.

Validation Results

- Comparison to original MATLAB version
  - Approximation technique used in each version listed

	MATLAB Single	VSIPL Double	VSIPL Single	Threshold
CNR Consistent	Yes	Yes	Yes	
CNR Δ	10^{-7}	10^{-14}	10^{-5}	$< 10^{-4}$
ECR	-101 dB	-130 dB	-98 dB	< -60 dB
ADMSE	10^{-7}	10^{-10}	10^{-6}	$< 10^{-3}$

Outline

- RF Clutter Simulation
- Validation Approach
- GPU VSIPL
- Precision Issues
- VSIPL Port, Optimization, and Results

VSIPL PORT

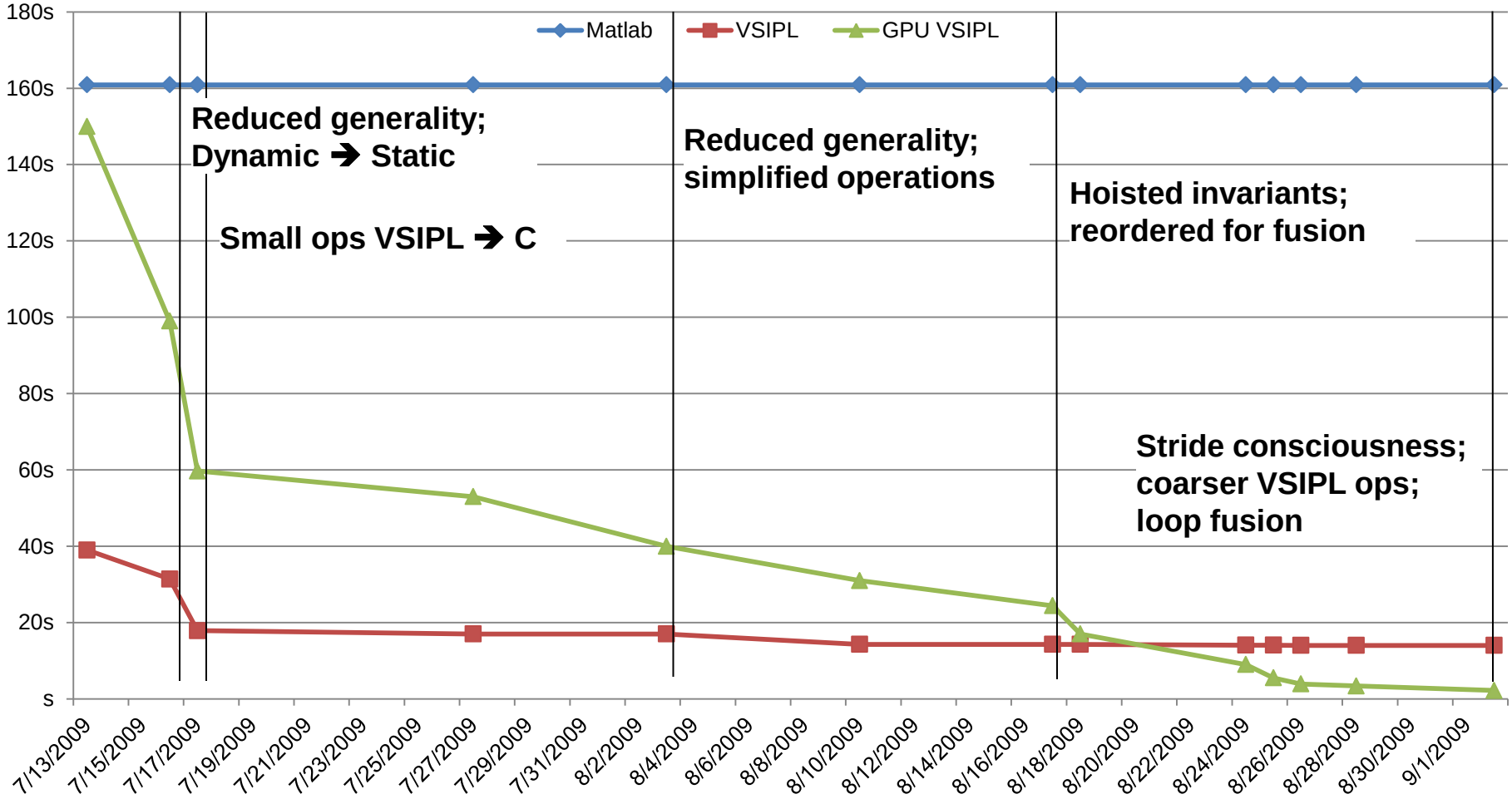
- **MATLAB to VSIPL port made easier due to VSIPL functions that emulate MATLAB operations**
- **Original MATLAB code very complex, particularly for radar novice**
  - First pass of the port was done with almost no attempts at optimizations
- **GPU transitioned required some additional changes**
  - Single vs Double precision issues
  - Time cost of operations differ TASP ←→ GPU

Optimization Issues

- **MATLAB code written for readability over speed**
  - Too many nested loops, operations involving small datasets
  - Many redundant calculations
- **Original code was very flexible, due to large user base**
  - Most optimizations required removing some generality
  - Assumptions need to be made about the scenario
- **Abstraction barrier issues**
  - Small operations less costly on CPU than GPU
  - Operation fusion, coarser operations, and leaving small things in C each helped

HPC Port – Performance

❖ **Optimization progression of single precision VSIPL:**



HPC Port – Performance

- **Performance Timing Results:**

Version	Runtime(s)	Speedup
MATLAB	162.5	1x
TASP VSIPL Double	20.9	7.8x
TASP VSIPL Single	14.0	11.6x
GPU VSIPL Single	2.2	73.8x
CUDA Native	TBD	

  - GTX 280/Q6600 TASP single core only