23 September 2010

Using GPU VSIPL & CUDA to Accelerate RF Clutter Simulation

2010 High Performance Embedded Computing Workshop

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

Radar will observe echo from object...

...as well as a strong return from the ground.

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

7,500 Hz

10,000 Hz

12,500 Hz

Targets at same range/Doppler as clutter will be obscured.
**Approach:** Sub-divide ground into number of unresolvable clutter patches and compute contribution of each.
RF Clutter Simulation

\[ x_m(t) = \sum_{l=1}^{L} A_{lm} s \left( t - \frac{2R_{lm}}{c} \right) \exp \left\{ i \frac{4\pi}{\lambda} R_{lm} \right\} \]

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_m(t) )</td>
<td>Clutter data for pulse ( m )</td>
</tr>
<tr>
<td>( s(t) )</td>
<td>Complex baseband radar waveform</td>
</tr>
<tr>
<td>( A_{lm} )</td>
<td>SNR of clutter patch ( l ) for pulse ( m )</td>
</tr>
<tr>
<td>( R_{lm} )</td>
<td>Range from radar to clutter patch ( l ) for pulse ( m )</td>
</tr>
<tr>
<td>( c )</td>
<td>Speed of wave propagation</td>
</tr>
<tr>
<td>( \lambda )</td>
<td>Radar wavelength</td>
</tr>
</tbody>
</table>

Radar clutter data is sum of delayed and phase shifted versions of radar waveform.
# RF Clutter Simulation

## Notional Parameters

<table>
<thead>
<tr>
<th></th>
<th>Air-to-Air</th>
<th>SAR Imaging (Air-to-Ground)</th>
<th>Our Test</th>
</tr>
</thead>
<tbody>
<tr>
<td># of Range Bins</td>
<td>200</td>
<td>1750</td>
<td>500</td>
</tr>
<tr>
<td># of Pulses</td>
<td>128</td>
<td>3000</td>
<td>8</td>
</tr>
<tr>
<td># of Clutter Patches</td>
<td>6,800 Rng x 96 Az = 6.5 x 10⁵</td>
<td>14,500 Rng x 26,812 Az = 3.8 x 10⁸</td>
<td>566 rng x 52 az = 29,432</td>
</tr>
</tbody>
</table>

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. transcendental functions
  - 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:

- \[ \text{CNR}_\Delta = \frac{\text{CNR}_M - \text{CNR}_T}{\text{CNR}_M} < 10^{-4} \]
- \[ \text{ECR} = 20 \log_{10} \left( \frac{\text{norm}(M(:) - T(:))}{\text{norm}(M(:))} \right) < -60\text{dB} \]
- \[ \text{ADMSE} = \text{Mean}( |\text{fft2}(M(:)) - \text{fft2}(T(:))|^2 ) < 10^{-3} \]
Outline

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

- 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
Outline

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

- **VSIPL versions compared to MATLAB version**

<table>
<thead>
<tr>
<th></th>
<th>VSIPL Double</th>
<th>VSIPL Single</th>
<th>Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CNR Consistent</strong></td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td><strong>CNR $\Delta$</strong></td>
<td>$10^{-1.6}$</td>
<td>$10^{-6}$</td>
<td>$&lt; 10^{-4}$</td>
</tr>
<tr>
<td><strong>ECR</strong></td>
<td>-152 dB</td>
<td>2.9 dB</td>
<td>$&lt; -60$ dB</td>
</tr>
<tr>
<td><strong>ADMSE</strong></td>
<td>$10^{-1.2}$</td>
<td>$10^4$</td>
<td>$&lt; 10^{-3}$</td>
</tr>
</tbody>
</table>
Single Precision

- Single precision errors caused by high dynamic range in platform to clutter patch range calculation:
  - \( d(\text{Platform} \rightarrow \text{clutter}) \gg \gg \gg d(\text{clutter patch} \rightarrow \text{clutter patch}) \)

- Solution: use far-field approximation technique
  - Double precision used to compute a base range
  - Single precision for sets of \( \Delta R \) 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 \( y \)

\[
R(x) = \|x - y\|
\]

Linear approximation near \( x_0 \)

\[
R(x) \approx R(x_0) + \left( \frac{x_0 - y}{\|x_0 - y\|} \right) \cdot (x - x_0)
\]

Unit vector from CPI center to clutter patch

Distance from center of scene, \( \epsilon \)

Distance travelled in direction orthogonal to “lines” of constant range

Quadratic Term

\[
\frac{1}{2} \left[ \frac{\|\epsilon\|^2}{\|x_0 - y\|} - \left( \epsilon \cdot \frac{x_0 - y}{\|x_0 - y\|} \right)^2 \right] \approx 0 \text{ for } \|\epsilon\| << \|x_0 - y\|
\]
Bounding Error

Approximation Error

\[
\left| R(x_0 + \epsilon) - \hat{R}(x_0 + \epsilon) \right| \leq \frac{1}{2} \frac{\|\epsilon\|^2}{\|x_0 - y\|}
\]

Case 1: Air-to-Air

128 pulses, 20 kHz PRF, 300 m/s velocity \(\rightarrow\) \(\|\epsilon\| < 1\text{m}\)
10 km Altitude \(\rightarrow\) \(\|x_0 - y\| < 10\text{km}\)

error < 50 \(\mu\text{m}\) < 0.06° phase at X band

Case 2: SAR

10 second dwell, 100 m/s velocity \(\rightarrow\) \(\|\epsilon\| < 500\text{m}\)
10 km Altitude \(\rightarrow\) \(\|x_0 - y\| < 10\text{km}\)

error < 12.5 m >> \(\lambda\) 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

<table>
<thead>
<tr>
<th>Approximation</th>
<th>MATLAB Single</th>
<th>VSIPL Double</th>
<th>VSIPL Single</th>
<th>Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNR Consistent</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>CNR Δ</td>
<td>$10^{-7}$</td>
<td>$10^{-14}$</td>
<td>$10^{-5}$</td>
<td>$&lt; 10^{-4}$</td>
</tr>
<tr>
<td>ECR</td>
<td>-101 dB</td>
<td>-130 dB</td>
<td>-98 dB</td>
<td>$&lt; -60$ dB</td>
</tr>
<tr>
<td>ADMSE</td>
<td>$10^{-7}$</td>
<td>$10^{-10}$</td>
<td>$10^{-6}$</td>
<td>$&lt; 10^{-3}$</td>
</tr>
</tbody>
</table>
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 transition required some additional changes
  - Single vs Double precision issues
  - Time cost of operations differ TASP ↔ GPU
- VSIPL needs “sample” function
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:

![Graph showing performance improvement over time](image)
HPC Port – Performance

- Optimization progression of single precision VSIPL:

- Reduced generality; Dynamic ➔ Static

Graph showing performance improvements over time for VSIPL and GPU VSIPL compared to Matlab.
## Optimization progression of single precision VSIPL:

<table>
<thead>
<tr>
<th>Date</th>
<th>Matlab</th>
<th>VSIPL</th>
<th>GPU VSIPL</th>
</tr>
</thead>
<tbody>
<tr>
<td>7/13/2009</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7/15/2009</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7/17/2009</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7/19/2009</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7/21/2009</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7/23/2009</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7/25/2009</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7/27/2009</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7/29/2009</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7/31/2009</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8/2/2009</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8/4/2009</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8/6/2009</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8/8/2009</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8/10/2009</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8/12/2009</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8/14/2009</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8/16/2009</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8/18/2009</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8/20/2009</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8/22/2009</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8/24/2009</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8/26/2009</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8/28/2009</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8/30/2009</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9/1/2009</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Notes:**
- Reduced generality; Dynamic ➔ Static
- Small ops VSIPL ➔ C
Optimization progression of single precision VSIPL:

- Reduced generality; Dynamic $\Rightarrow$ Static
- Small ops VSIPL $\Rightarrow$ C

Timeline:
- 180s
- 140s
- 160s
- 120s
- 60s
- 40s
- 20s
- Reduced generality; simplified operations
HPC Port – Performance

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

![Graph showing optimization progression](image)

- **Reduced generality; Dynamic ➔ Static**
- **Small ops VSIPL ➔ C**
- **Reduced generality; simplified operations**
- **Hoisted invariants; reordered for fusion**
Optimization progression of single precision VSIPL:

- Reduced generality; Dynamic ➔ Static
- Small ops VSIPL ➔ C
- Reduced generality; simplified operations
- Hoisted invariants; reordered for fusion
- Stride consciousness; coarser VSIPL ops; loop fusion
HPC Port – Performance

Performance Timing Results:

<table>
<thead>
<tr>
<th>Version</th>
<th>Runtime(s)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>MATLAB</td>
<td>162.5</td>
<td>1x</td>
</tr>
<tr>
<td>TASP VSIPPL Double</td>
<td>20.9</td>
<td>7.8x</td>
</tr>
<tr>
<td>TASP VSIPPL Single</td>
<td>14.0</td>
<td>11.6x</td>
</tr>
<tr>
<td>GPU VSIPPL Single</td>
<td>2.2</td>
<td>73.8x</td>
</tr>
<tr>
<td>CUDA Native</td>
<td>1.3</td>
<td>125x</td>
</tr>
</tbody>
</table>

- GTX 480/Q6600 TASP single core only