Fast Hybrid QR-Cholesky Factorization and Tuning Techniques for STAP Algorithm Implementation on Graphics Processor Architectures for Embedded Systems.

Dr. Dennis Healy, DARPA, Dr. Dennis C. Braunreiter, Dr. Jackie Sillaci, David Boe, Jeremy Furtek, SAIC, Prof. Xaiobai Sun, Duke University. Contact information: braunreiterd@saic.com, 858-826-5544.

Emerging capabilities in stream and multi-core computation along with high speed memory bandwidths in commercial graphics processor (GPU) architectures are enabling breakthrough low cost and low power teraflop computing solutions to DoD embedded computing challenges. Under the DARPA MTO STAP-BOY program, SAIC and Duke University have been developed mappings of complex signal processing algorithms to GPU architectures in cooperation with commercial graphics processor companies, with the focus on STAP applications for radar adaptive beamforming. The first phase of the program was focused on benchmarking algorithms mapped to GPUs based on emergent stream computing and graphics shader APIs. The conclusions reached in the STAP-BOY phase one program is that the compute architecture and stream and graphics APIs lead to order(s) of magnitude power/watt/$ improvements over CPU and DSP solutions, with high productivity in application development.

Under the DARPA STAP-BOY program, the implementation of a prototype radar STAP algorithm in work done under the DARPA STAP-BOY program was completed, using the close to the metal, stream programming (Peakstream, ATI-CTM, and NVIDIA), and traditional graphics APIs, and trade implementations of the STAP algorithm for fast range adaptive STAP weight updates. In addition to the weight computation, that beamforming is also a natural match for graphics architectures. With the memory and multi-core architecture of the graphics processor, a single chip graphics chip solution is available for a large class of STAP airborne radar applications with application to similar size problems with different sensor modalities. In our study, we focused particularly on the case of 128 degrees of freedom for a 16-element airborne antenna array, with 16 spatial taps, 2 time taps and 4 Doppler taps. Our initial results are shown in figure 1. Finally, our presentation shows the productivity enhancement achieved by exploit the graphics processor memory and compute cores. In our phase one work, we focused on fast range adaptive weight updates that

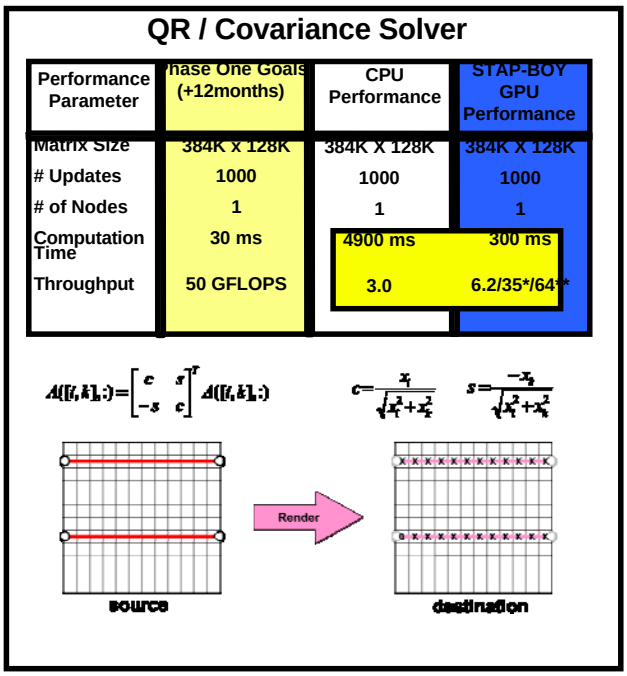


Figure 1. Initial STAP-BOY QR Benchmark

were updated once per range gate in he prototype radar airborne surveillance application that requires the covariance matrix to be factorized once per range gate with 128 degrees of freedom.. In addition, in our presentation, we will show the results in this briefing that trade QR factorization implementations utilizing low cost commercial API and tools in an integrated development environment developed by SAIC to rapidly achieve high throughput computational performance.

A key innovation in our development of the STAP algorithm mapping to the graphics processor architecture is the mathematical reorganization of the STAP algorithm to highest computational performance. In our approach, we will also show the advantage of the batch processing formulation of weight updates on the GPU as part of the solver that maps naturally to the GPU cores. The key approach to the batch processing formulation is to realize that a core factorization can be done for multiple updates and low ranks update for additional solutions. The size of the core matrix factorization is tuned to the performance of the graphics processor chip with software tuning tools that have been developed by SAIC.

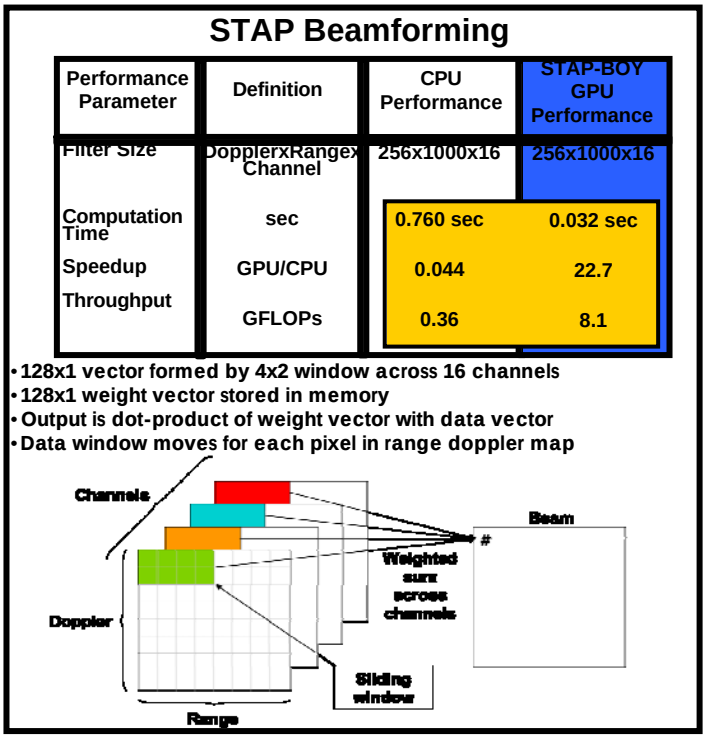
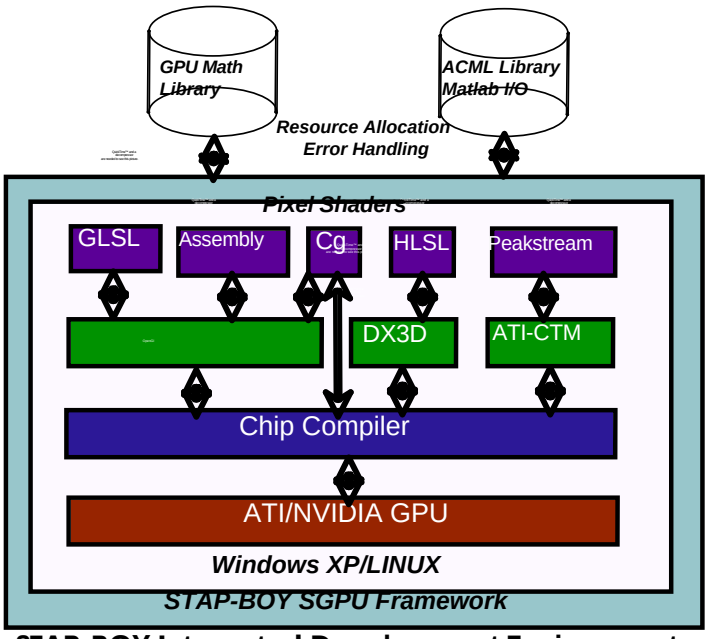


Figure 2. Initial STAP-Beam Forming Performance

In addition to the covariance matrix factorization for weight solution, the 128 weights for our prototype problem are stored for each range gate after computation, retrieved from graphics memory, and applied without leaving the graphics processor memory. In order to benchmark the fully adaptive weight solution problem we show that the memory bandwidth from the processor cores to graphics memory can achieve over 20Gbytes/second when the number of computations exceed 4 times the number of memory fetches. The parallel 4x1 floating point vector processor architecture in the graphics shader responsible for RGBA texturing provides a convenient approach to breaking up feedforward computational problems into factors or 4. Our initial results for the fully adaptive beamforming application are shown in figure 3 using the graphics API, and over 64 GFLOPs based on initial benchmarks with the stream programming APIs when the fully adaptive beamforming problem is treated as a matrix-matrix multiply.



STAP-BOY Integrated Development Environment

- **100% COTS and/or open source**
- **42,000 lines of code**
- **Cross platform suite of libraries**
- **Automation of common tasks**
- **Utilities developed by College Interns**

Figure 3. STAP-BOY SGPU Development Environment

A challenge that faces mapping new algorithms to the multi-core graphics processor applications is ease of algorithm mapping and software development. In the phase one STAP-BOY program, SAIC developed over 40,000 lines of code and an development framework that interfaces with the existing commercial APIs and software packages that enables cross platform and cross operating system compilation as shown in figure 3. With the application development framework developed in STAP-BOY, productivity is shown that over 90% of the computational throughput performance in implementation can be achieved within a 2-week development cycle.

In phase two of the DARPA STAP-BOY program, we are focusing on the mapping of large scale end-to-end video and 3D radar surveillance applications in a multi-GPU embedded computing architecture for embedded applications. In addition to our presentation, we will also have a demonstration of 3D object recognition on prototype STAP-BOY GPU hardware and comparison to a CPU based implementation..