The HPEC Challenge Benchmark Suite

Ryan Haney, Theresa Meuse, Jeremy Kepner and James Lebak

Massachusetts Institute of Technology
Lincoln Laboratory

HPEC 2005

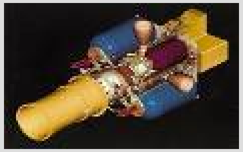
This work is sponsored by the Defense Advanced Research Projects Agency under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

Acknowledgements

- Lincoln Laboratory PCA Team
  - Matthew Alexander
  - Jeanette Baran-Gale
  - Hector Chan
  - Edmund Wong
- Shomo Tech Systems
  - Marti Bancroft
- Silicon Graphics Incorporated
  - William Harrod
- Sponsor
  - Robert Graybill, DARPA PCA and HPCS Programs
- Code adapted from Soumekh, Mehrdad, *Synthetic Aperture Radar Signal Processing with Matlab Algorithms*, Wiley, 1999

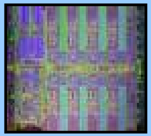
Motivation

Advanced Sensor Platforms

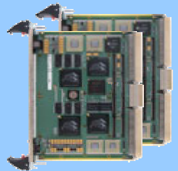


Processor and System Architectures

Single Processor Element



Tiled Processors



Multi-computers

Super-computers

System Analysis and Design

Implement Benchmarks

- Design
- Code
- Tune

Measure Performance

- Throughput
- Power
- Stability

Design System

- Choose components
- Hardware size
- Required software performance

Challenge: Provide benchmarks that test a system at the kernel and multi-processor levels

HPEC Challenge Benchmark Suite

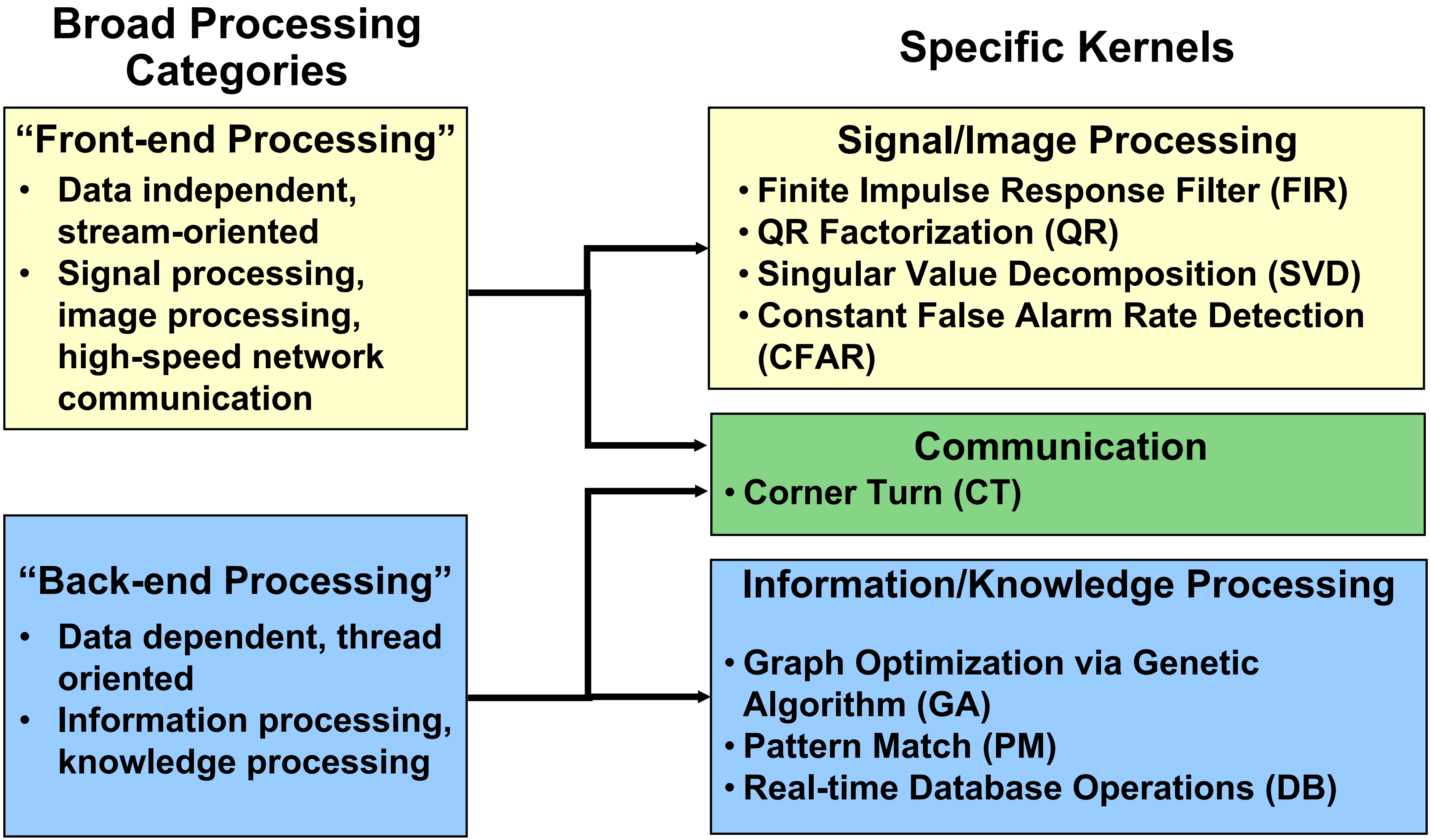
- PCA program kernel benchmarks
  - Single-processor operations
  - Drawn from many different DoD applications
  - Represent both “front-end” signal processing and “back-end” knowledge processing

- HPCS program Synthetic SAR benchmark
  - Multi-processor compact application
  - Representative of a real application workload
  - Designed to be easily scalable and verifiable

Outline

- Introduction
- Kernel Level Benchmarks
  - Kernel Overview
  - Kernel Architecture
  - Generic vs. Optimized Results
- SAR Benchmark
- Release Information
- Summary

Kernel Benchmark Selection



Broad Processing Categories

“Front-end Processing”

- Data independent, stream-oriented
- Signal processing, image processing, high-speed network communication

“Back-end Processing”

- Data dependent, thread oriented
- Information processing, knowledge processing

Specific Kernels

Signal/Image Processing

- Finite Impulse Response Filter (FIR)
- QR Factorization (QR)
- Singular Value Decomposition (SVD)
- Constant False Alarm Rate Detection (CFAR)

Communication

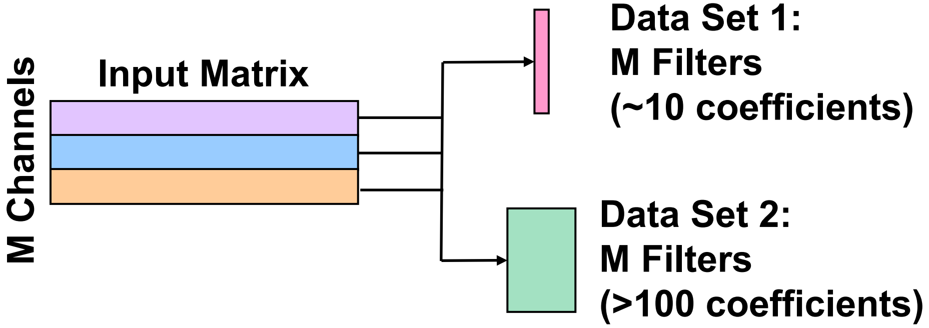
- Corner Turn (CT)

Information/Knowledge Processing

- Graph Optimization via Genetic Algorithm (GA)
- Pattern Match (PM)
- Real-time Database Operations (DB)

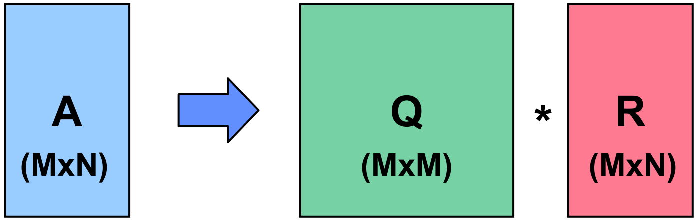
Signal and Image Processing Kernels

FIR



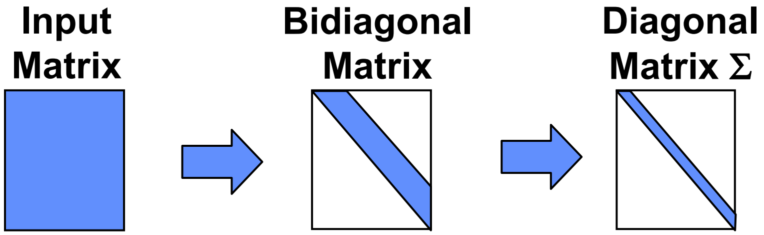
- Bank of filters applied to input data
- FIR filters implemented in time and frequency domain

QR



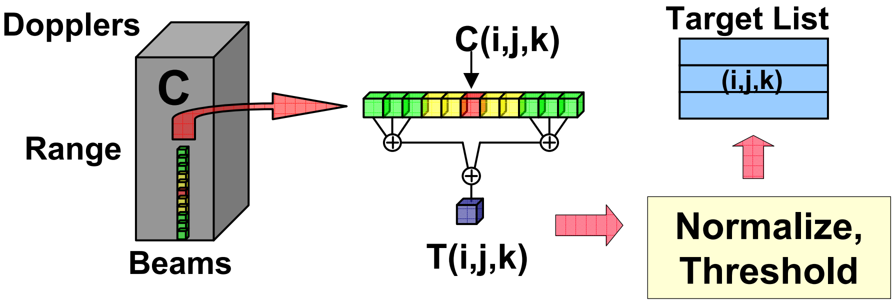
- Computes the factorization of an input matrix, A=QR
- Implementation uses Fast Givens algorithm

SVD



- Produces decomposition of an input matrix, $X=U\Sigma V^H$
- Classic Golub-Kahan SVD implementation

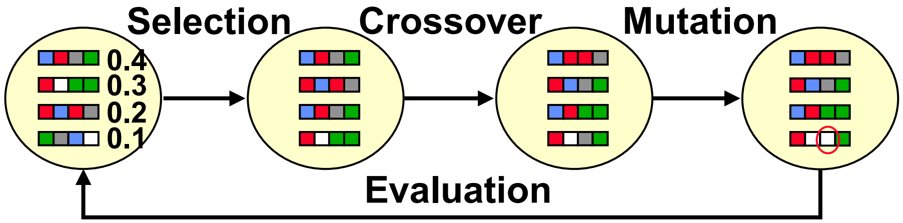
CFAR



- Creates a target list given a data cube
- Calculates normalized power for each cell, thresholds for target detection

Information and Knowledge Processing Kernels

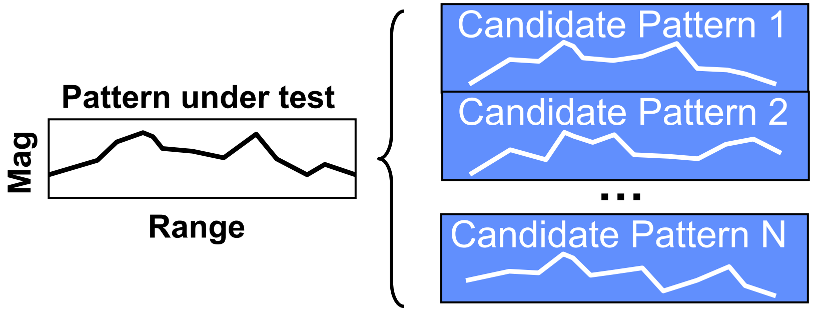
Genetic Algorithm



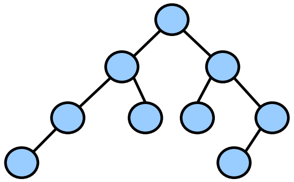
- Evaluate each chromosome
- Select chromosomes for next generation
- Crossover: randomly pair up chromosomes and exchange portions
- Mutation: randomly change each chromosome

Pattern Match

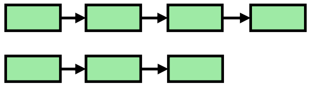
- Compute best match for a pattern out of set of candidate patterns
  - Uses weighted mean-square error



Database Operations



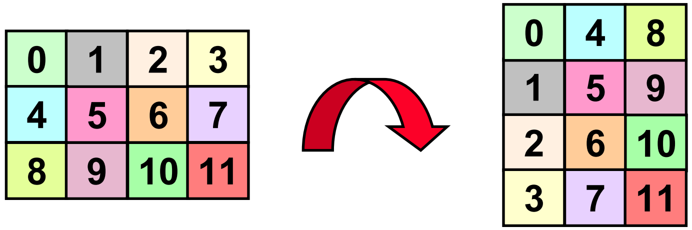
Red-Black Tree
Data Structure



Linked List
Data Structures

- Three generic database operations:
  - search: find all items in a given range
  - insert: add items to the database
  - delete:remove item from the database

Corner Turn

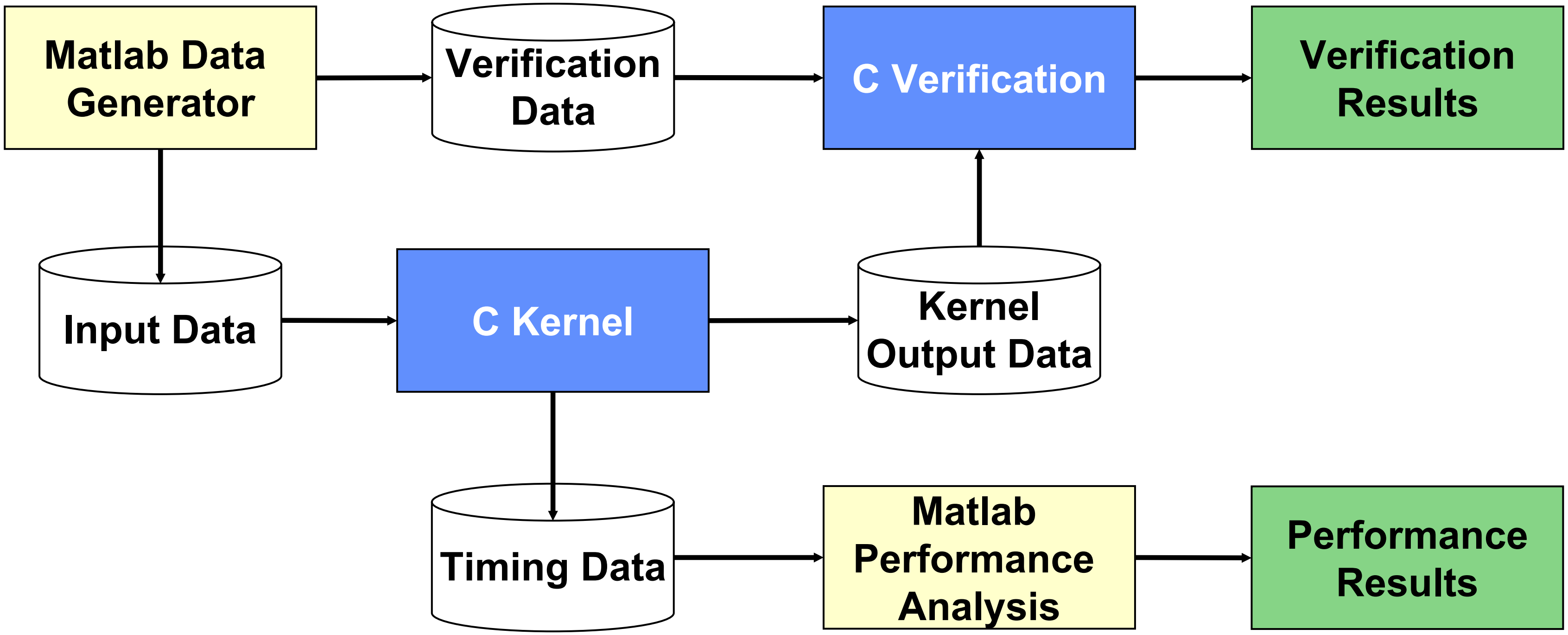


0	1	2	3
4	5	6	7
8	9	10	11

0	4	8
1	5	9
2	6	10
3	7	11

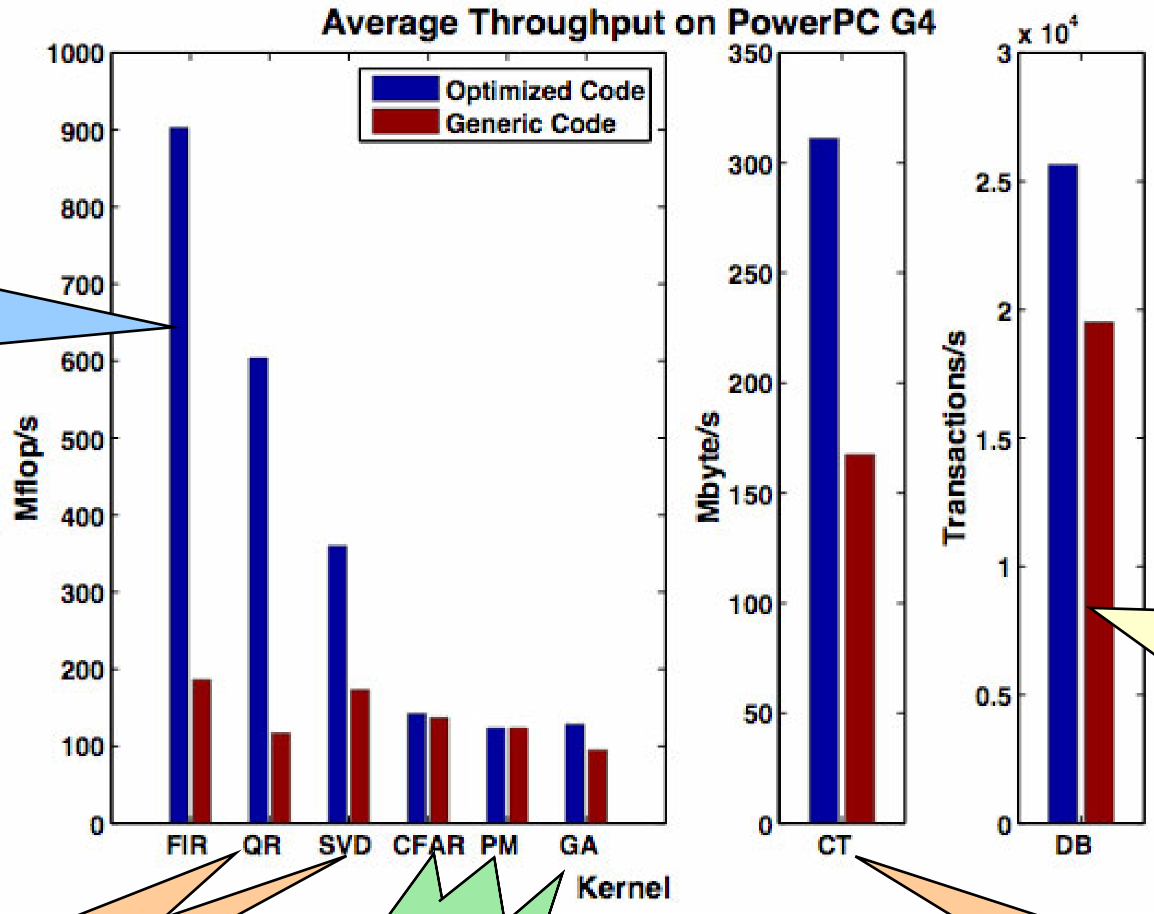
- Memory rearrangement of matrix contents
  - Switch from row to column major layout

Kernel Benchmark Architecture



- Kernels written in ANSI C
  - Portable across UNIX based platforms
- Sample data sets provided
- Generation of user defined data sets and sizes possible using Matlab

Generic vs. Optimized Kernel Benchmark Results



Optimized code outperforms baseline in FIR due to use of VSIPL libraries.

Optimized DB code uses specialized memory management routines.

QR and SVD optimized code outperforms baseline because of AltiVec use.

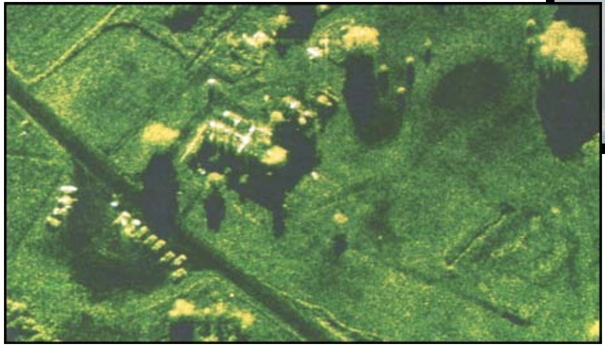
Results similar for CFAR, PM, and GA. Minimal optimizations were made.

Optimized Corner Turn code uses AltiVec intrinsics.

Outline

- Introduction
- Kernel Level Benchmarks
- SAR Benchmark
  - Overview
  - System Architecture
  - Computational Components
- Release Information
- Summary

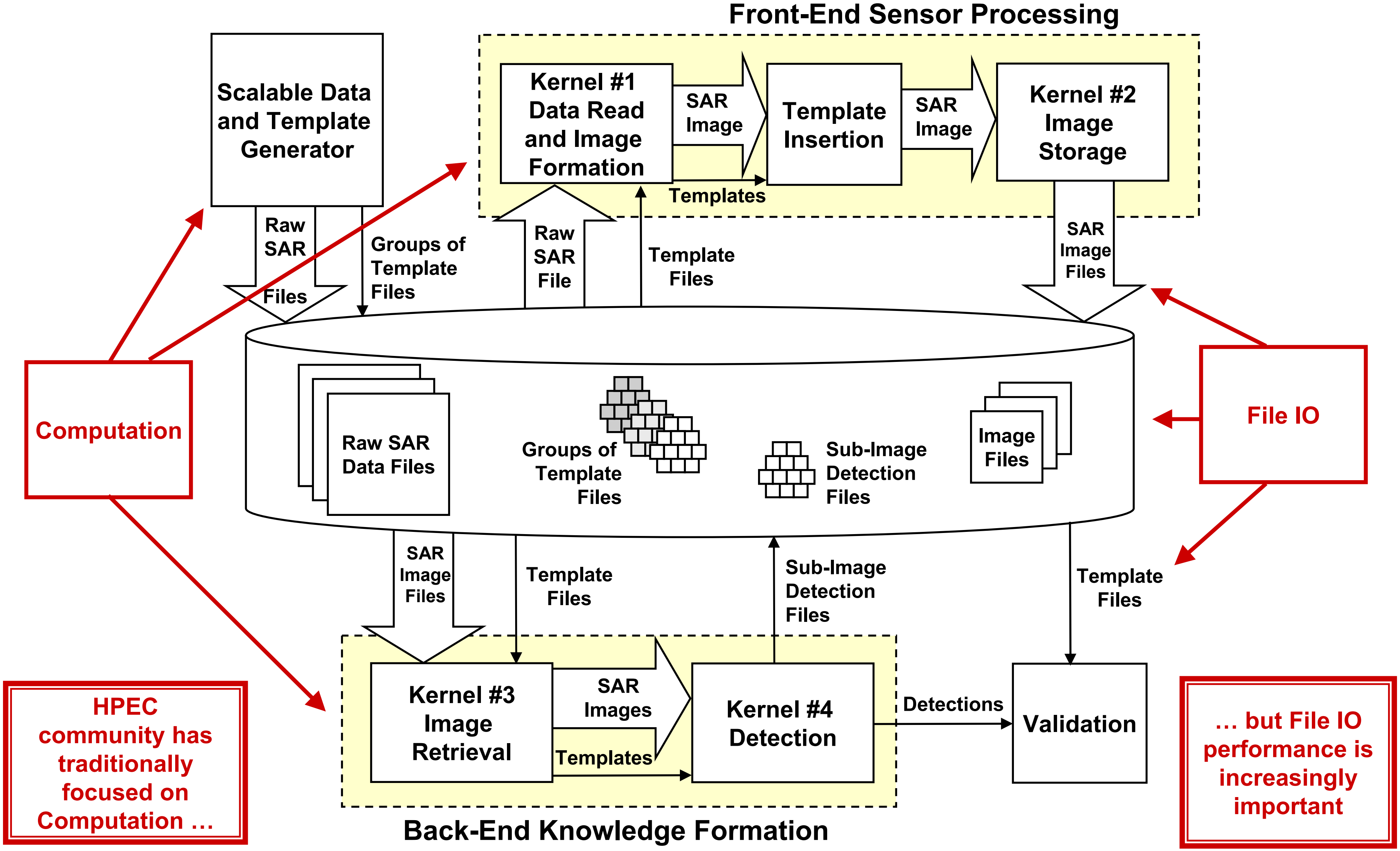
Spotlight SAR System



- Principal performance goal: Throughput
  - Maximize rate of results
  - Overlapped IO and computing

- Intent of Compact App:
  - Scalable
  - High Compute Fidelity
  - Self-Verifying

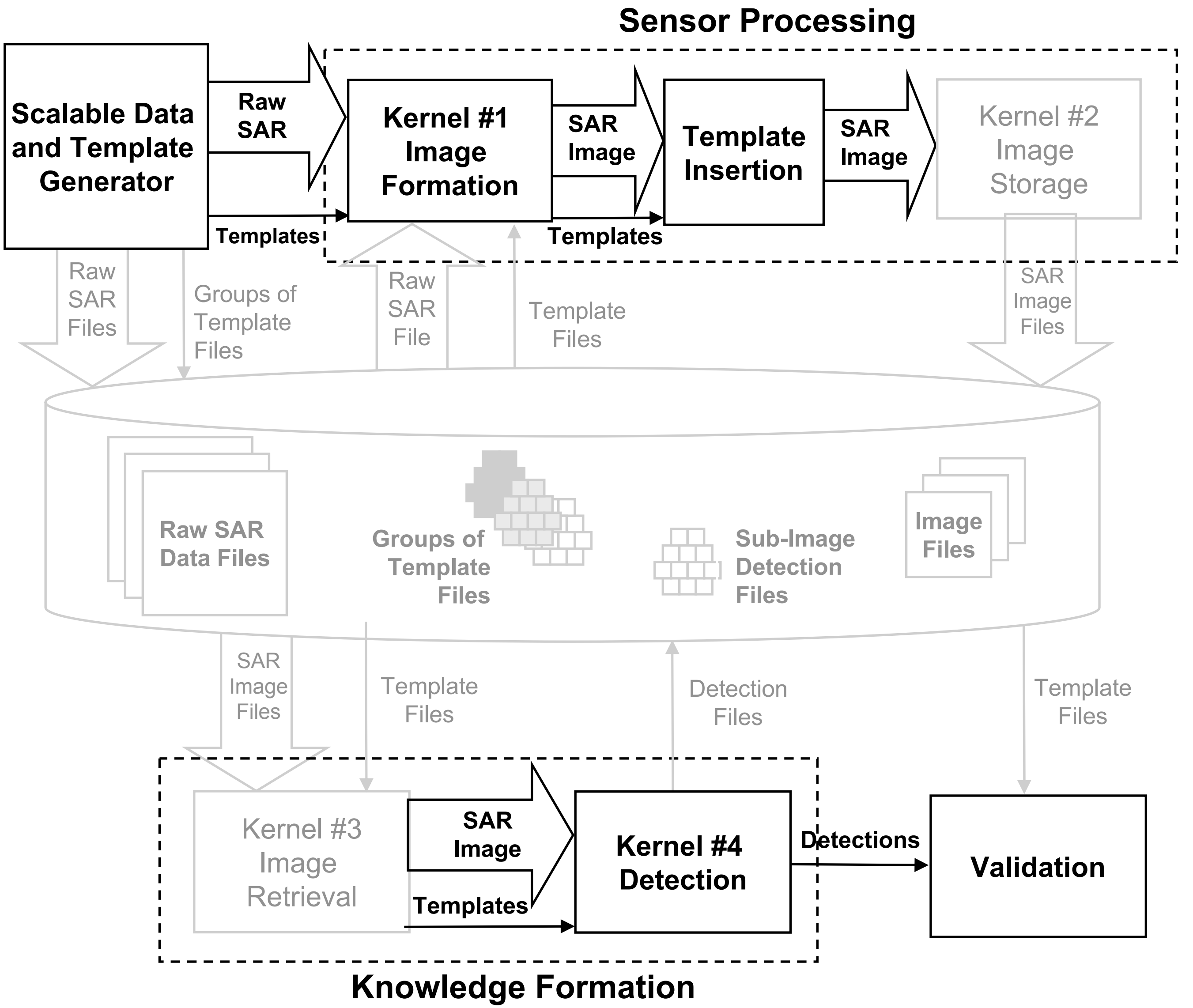
SAR System Architecture



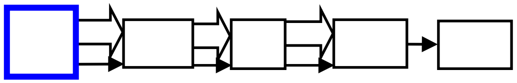
HPEC community has traditionally focused on Computation …

… but File IO performance is increasingly important

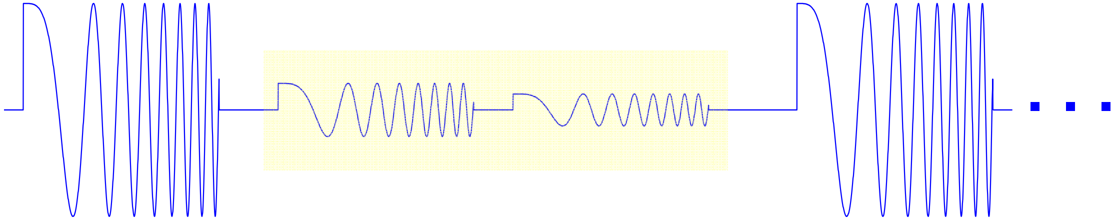
Data Generation and Computational Stages

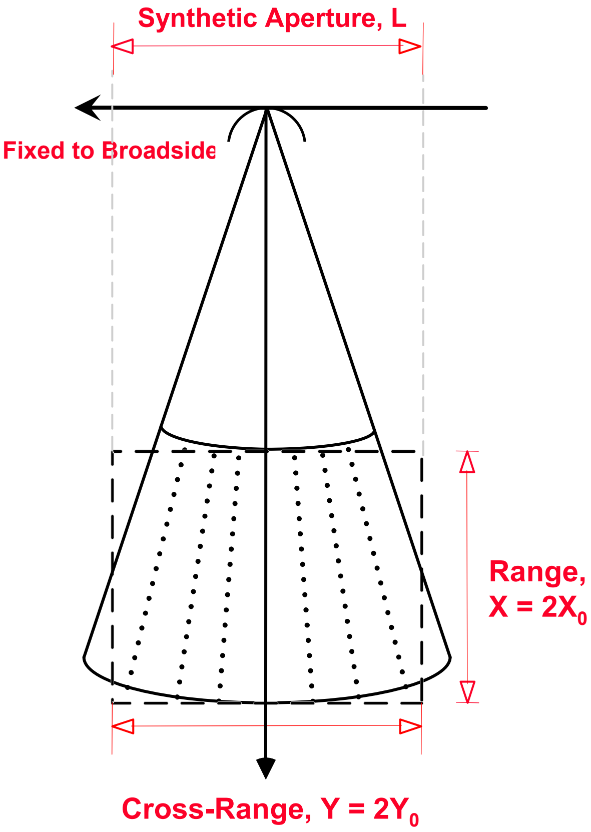


SAR Overview



- Radar captures echo returns from a ‘swath’ on the ground
- Notional linear FM chirp pulse train, plus two ideally non-overlapping echoes returned from different positions on the swath





- Summation and scaling of echo returns realizes a challengingly long antenna aperture along the flight path

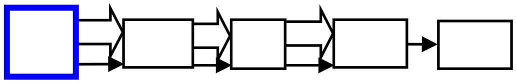
delayed transmitted SAR waveform

$$s(t,u)=\sum_{pulses}\sum_{swath}\alpha(n,m)\,p\big(t-\tau(n,m)\big)$$

received ‘raw’ SAR

reflection coefficient scale factor, different for each return from the swath

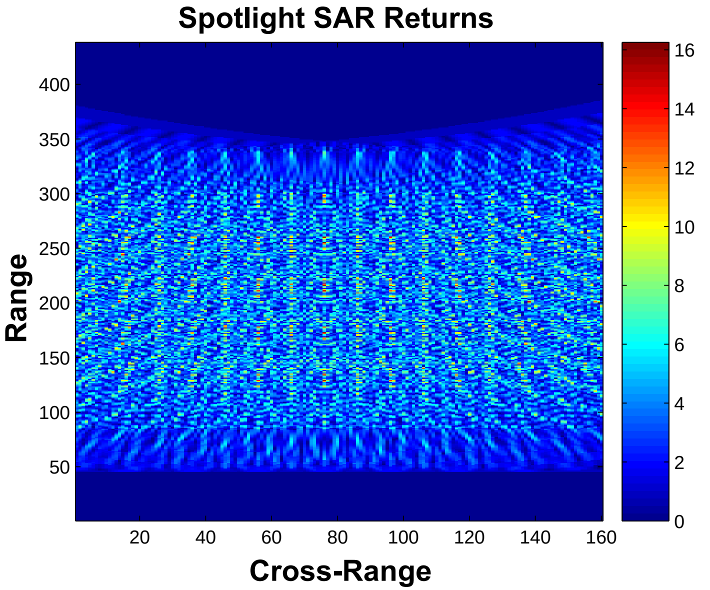
Scalable Synthetic Data Generator



- Generates synthetic raw SAR complex data

- Data size is scalable to enable rigorous testing of high performance computing systems
  - User defined scale factor determines the size of images generated

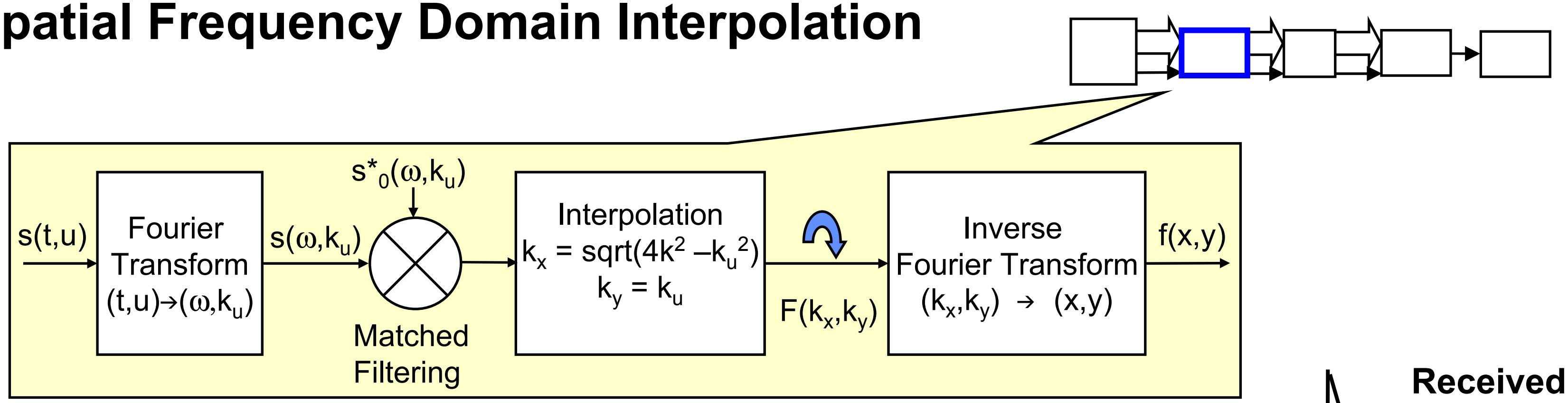
- Generates 'templates' that consist of rotated and pixelated capitalized letters



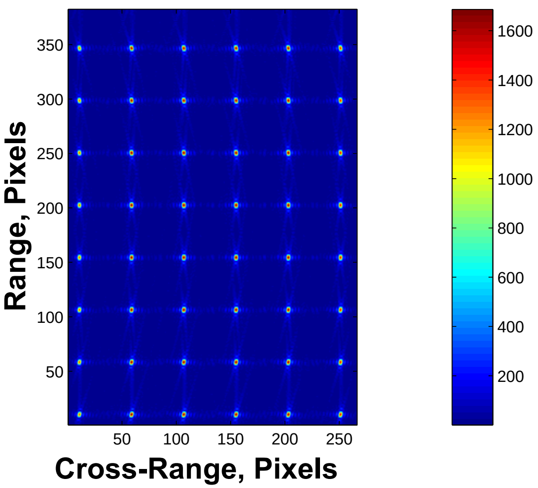
Source Code adapted from Soumekh, 1999.

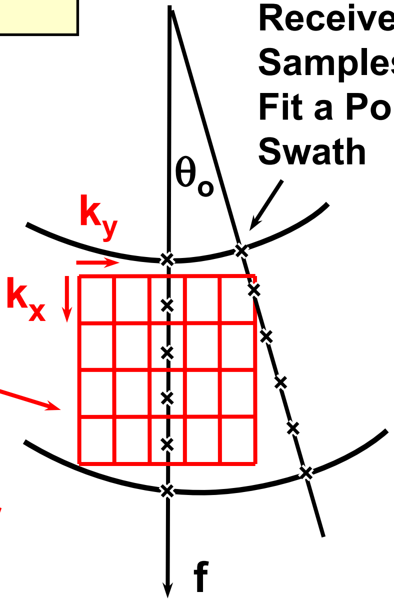
Kernel 1 — SAR Image Formation

Spatial Frequency Domain Interpolation


$s(t,u)$
Fourier Transform
$(t,u) \rightarrow (\omega,k_u)$
$s(\omega,k_u)$
$s^*_0(\omega,k_u)$
Matched Filtering
Interpolation
$k_x = \mathrm{sqrt}(4k^2 - k_u^2)$
$k_y = k_u$
$F(k_x,k_y)$
Inverse Fourier Transform
$(k_x,k_y) \rightarrow (x,y)$
$f(x,y)$

Spotlight SAR Reconstruction





Received Samples Fit a Polar Swath

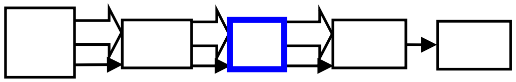
Processed Samples Fit a Rectangular Swath

Source Code adapted from Soumekh, 1999.

Template Insertion

(untimed)

- **Inserts rotated pixelated capital letter templates into each SAR image**
  - Non-overlapping locations and rotations
  - Randomly selects 50%
  - Used as ideal detection targets in Kernel 4



If inserted with
%100 Templates

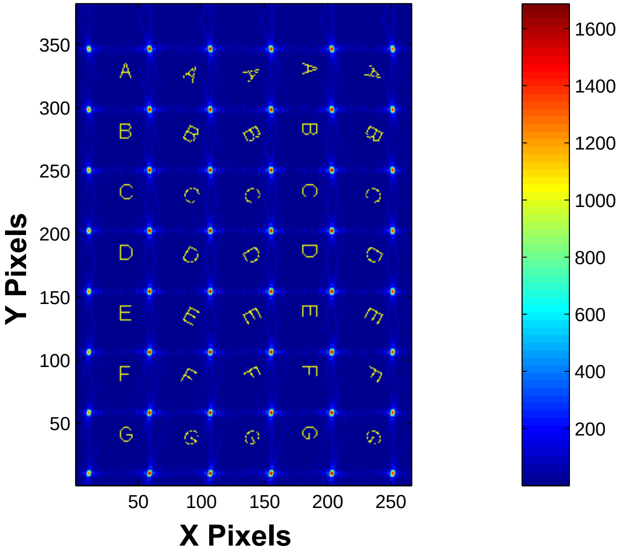
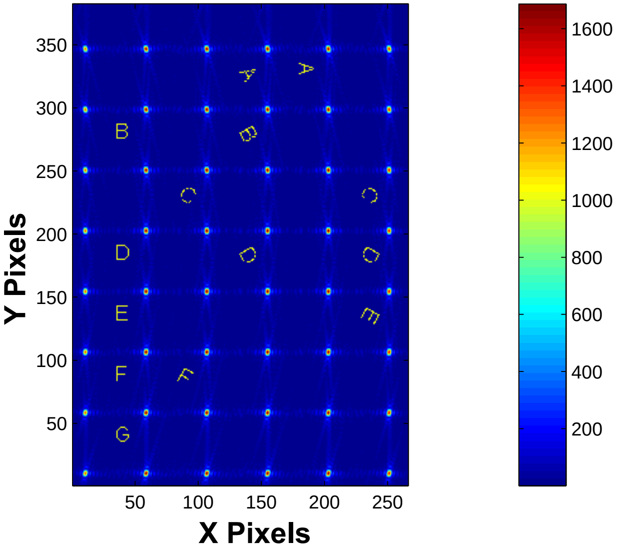
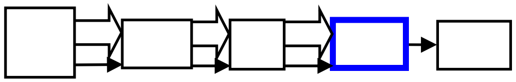


Image only inserted with
%50 random Templates



Kernel 4 — Detection

- Detects targets in SAR images
  1. Image difference
  2. Threshold
  3. Sub-regions
  4. Correlate with every template → max is target ID



- Computationally difficult
  – Many small correlations over random pieces of a large image
- 100% recognition no false alarms

Image A

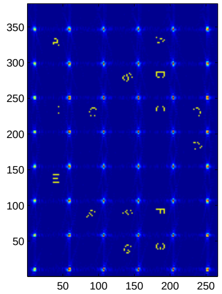


Image B

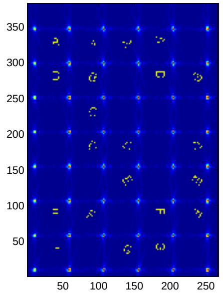
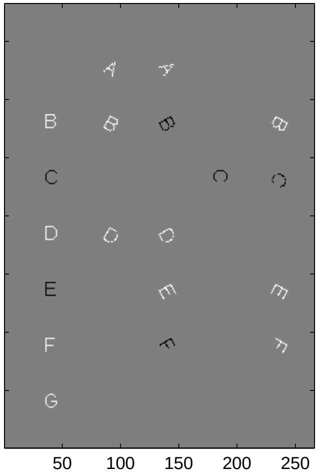
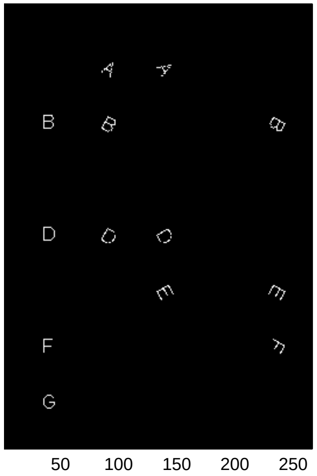


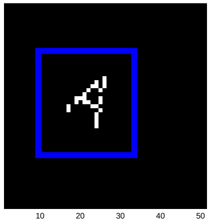
Image Difference

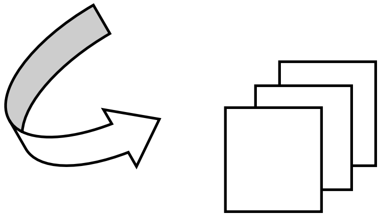


Thresholded Difference

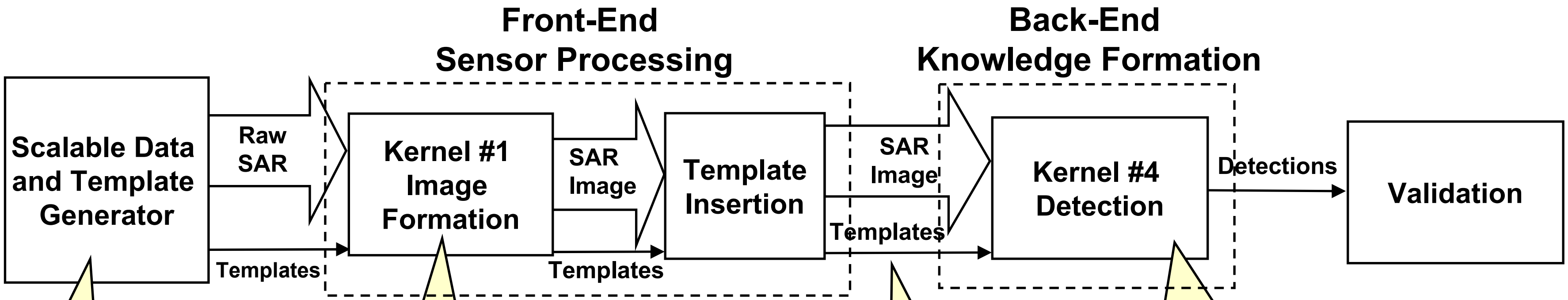


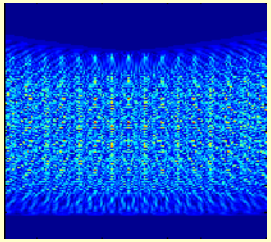
Sub-region



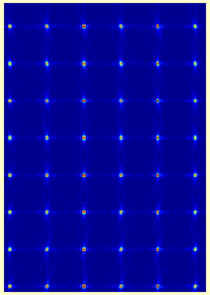


Benchmark Summary and Computational Challenges

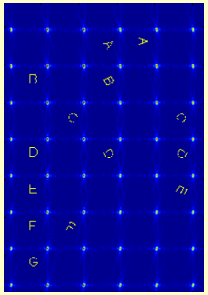




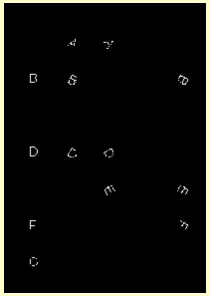
- Scalable synthetic data generation

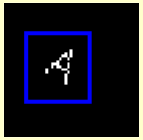


- Pulse compression
- Polar Interpolation
- FFT, IFFT (corner turn)



- Sequential store
- Non-sequential retrieve
- Large & small IO





- Large Images difference & Threshold

- Many small correlations on random pieces of large image

Outline

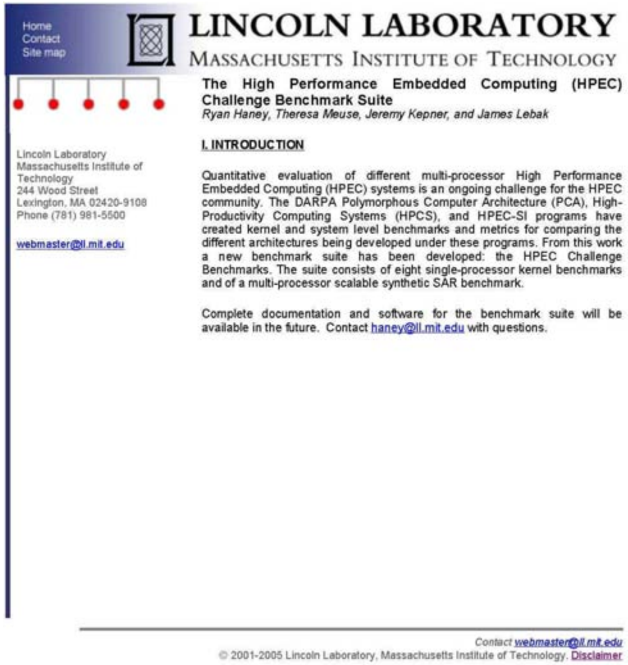
- Introduction
- Kernel Level Benchmarks
- SAR Benchmark
- ➡ Release Information
- Summary

HPEC Challenge Benchmark Release

- http://www.ll.mit.edu/HPECChallenge/
  - Future site of documentation and software

- Initial release is available to PCA, HPCS, and HPEC SI program members through respective program web pages
  - Documentation
  - ANSI C Kernel Benchmarks
  - Single processor MATLAB SAR System Benchmark

- Complete release will be made available to the public in first quarter of CY06



LINCOLN LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

The High Performance Embedded Computing (HPEC) Challenge Benchmark Suite
Ryan Haney, Theresa Meuse, Jeremy Kepner, and James Lebak

Lincoln Laboratory
Massachusetts Institute of Technology
244 Wood Street
Lexington, MA 02420-9108
Phone (781) 981-5500

webmaster@ll.mit.edu

I. INTRODUCTION

Quantitative evaluation of different multi-processor High Performance Embedded Computing (HPEC) systems is an ongoing challenge for the HPEC community. The DARPA Polymorphous Computer Architecture (PCA), High-Productivity Computing Systems (HPCS), and HPEC-SI programs have created kernel and system level benchmarks and metrics for comparing the different architectures being developed under these programs. From this work a new benchmark suite has been developed: the HPEC Challenge Benchmarks. The suite consists of eight single-processor kernel benchmarks and of a multi-processor scalable synthetic SAR benchmark.

Complete documentation and software for the benchmark suite will be available in the future. Contact haney@ll.mit.edu with questions.

Contact webmaster@ll.mit.edu
© 2001-2005 Lincoln Laboratory, Massachusetts Institute of Technology. Disclaimer

Summary

- The HPEC Challenge is a publicly available suite of benchmarks for the embedded space
  - Representative of a wide variety of DoD applications
- Benchmarks stress computation, communication and I/O
- Benchmarks are provided at multiple levels
  - Kernel: small enough to easily understand and optimize
  - Compact application: representative of real workloads
  - Single-processor and multi-processor
- For more information, see http://www.ll.mit.edu/HPECChallenge/

Backup Slides

Kernel Data Set Summary

Kernel	Parameter	Set 1	Set 2	Set 3	Set 4
FIR	Filters	64	20		
	Coefs	128	12		
	Vec Length	4096	1024		
QR/SVD	Matrix size	500x100	180x60	150x150	
CFAR Detection	Beams	16	48	48	16
	Range gates	64	3500	1909	9900
	Dopplers	24	128	64	16
Corner Turn	Matrix size	50x5000	750x5000		
Pattern Match	Pattern length	64	128		
	Number of patterns	72	256		
Database	Total records	500	102,400		
	Ops per cycle	440	700		
Genetic algorithm	Genes	8	96	5	10
	Chromosomes	50	150	50	150