Optimizing An Innovative SAR Post-Processing Algorithm for Multi-Core Processors

Peter Carlston,
Embedded Computing Division
Intel Corporation

Dave Murray,
N.A. Software Ltd.

Legal Disclaimer

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or by visiting Intel's Web Site.

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, visit Intel Performance Benchmark Limitations

All information provided related to future Intel products and plans is preliminary and subject to change at any time, without notice. All dates provided are subject to change without notice. Intel may make changes to specifications and product descriptions at any time, without notice.

Intel, Intel logo, Intel Core, and Xeon are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

64-bit computing on Intel architecture requires a computer system with a processor, chipset, BIOS, operating system, device drivers and applications enabled for Intel® 64 architecture. Performance will vary depending on your hardware and software configurations. Consult with your system vendor for more information.

The code names Tigertown, Dunnington, and Nehalem, resented in this document are only for use by Intel to identify products, technologies, or services in development, that have not been made commercially available to the public, i.e., announced, launched or shipped. They are not "commercial" names for products or services and are not intended to function as trademarks.

* Other names and brands may be claimed as the property of others.

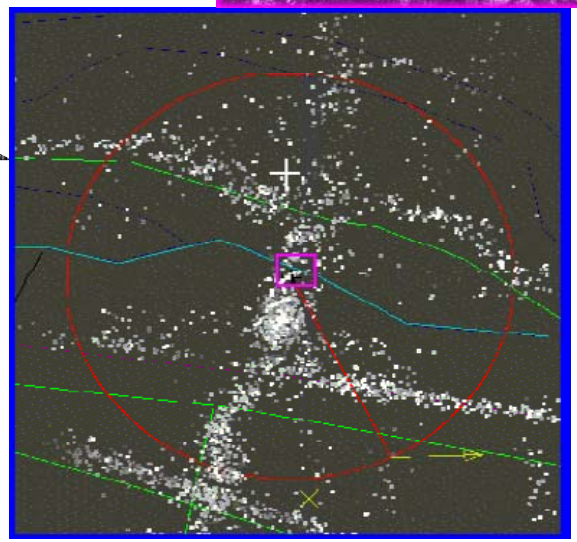
Other vendors are listed by Intel as a convenience to Intel's general customer base, but Intel does not make any representations or warranties whatsoever regarding quality, reliability, functionality, or compatibility of these devices. This list and/or these devices may be subject to change without notice.

Optimizing SARMTI for Multi-Core Processors

- A major US defense contractor asked NA Software, Ltd*. and Intel's Embedded Computing Division to multi-thread and evaluate an innovative SAR post-processing algorithm
  - "SARMTI" developed by Dr. Chris Oliver, CBE, of InfoSAR*
  - Identifies slow-/fast-moving targets from existing SAR data – no need for MTI radar
- Goals:
  - Phase 1: How will the original, serial algorithm scale across multiple processor cores when it is multi-threaded?
  - Phase 2: Can moving targets be detected, motions derived, and objects registered on a background SAR image in near real-time?

Existing SAR and MTI Algorithms

- SAR
  - Excellent along-track resolution -- ~1m
  - Moving targets degrade result
- MTI
  - Poor along-track resolution >100m
  - Good at identifying fast moving targets, but slow targets and ground clutter degrade result
  - Current “integrated” systems switch between SAR and MTI – require human analysis, several passes



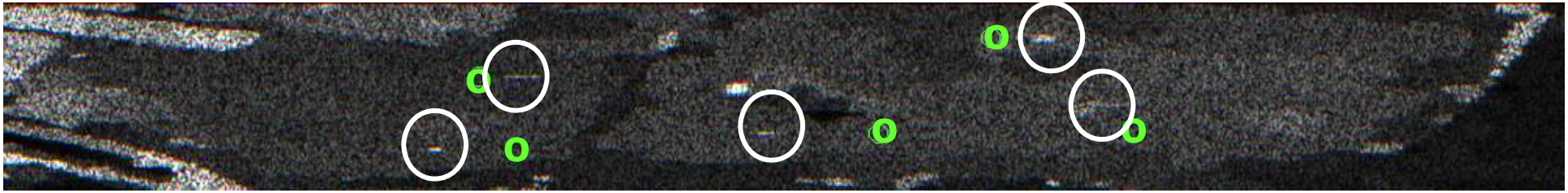
Existing SAR and MTI algorithms are both sub-optimal

New SARMTI Algorithm from InfoSAR*

Moving Target Position
- Actual
- SAR calc.
- SARMTI calc.

Existing SAR:

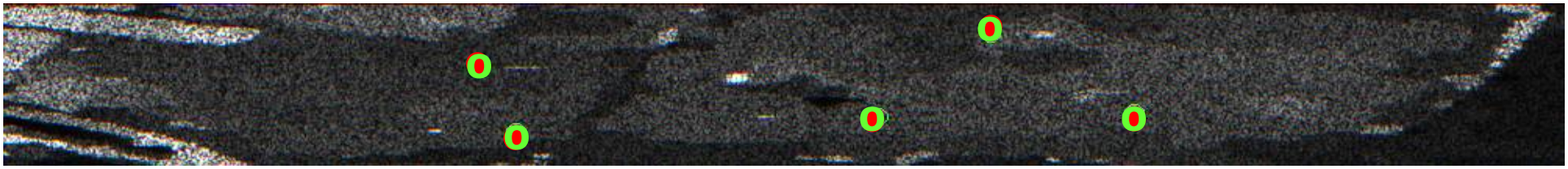
Moving targets are both shifted and blurred



After SARMTI applied to same SAR data: Targets detected (red circles) in correct positions within one SAR resolution cell

Corresponding motions are also derived

- Across-track acceleration and along-track velocity in addition to across-track velocity measured in typical MTI systems



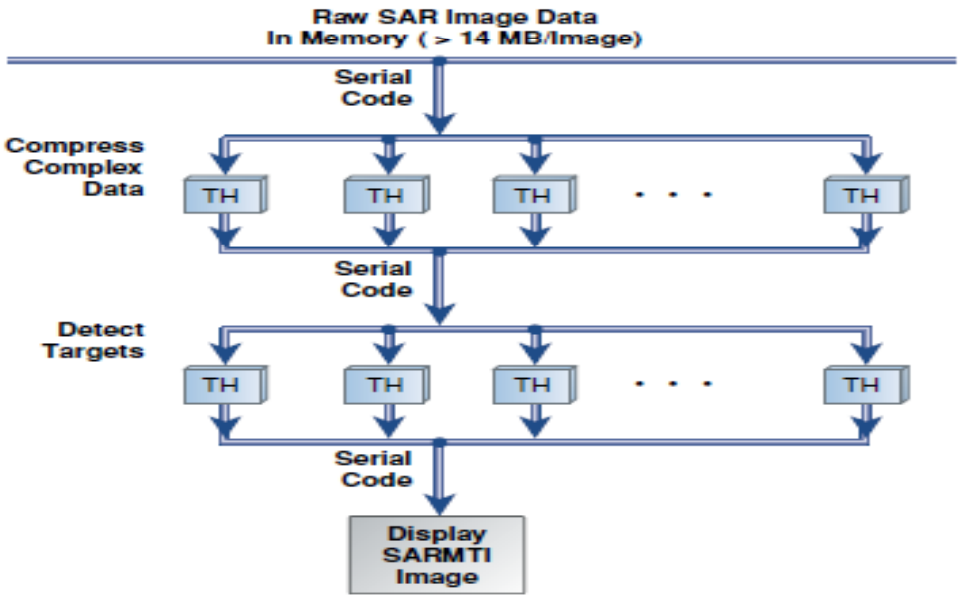
Multi-Threading: Round 1

- PPC code ported to IA
- Gnu* 'gprof' used to determine time spent in each function ("hot spots")

Round 1 gprof Results	
Time %	**Operation**
30.48	DETECTION.
53.07	FFTW Calls
11.22	OTHER COMPRESSION.
2.04	DATA EXTRACTION
96.81 %	**Total**

- Multi-threaded compression and detection phases
- Restructured calls to most used target detection functions
  - Only a few calls—all in initial stage

SARMTI: Multi-Threaded Structure



- Raw SAR image (>14MBytes) loaded into memory
- Serial processing for set-up, synchronization and image display)
- Data tiles processed independently on each core (thread)
  - Avoid ‘cache thrash’: Be careful to use localized memory for each thread
  - Processor (Core) Affinity vs letting Linux dynamically place each process on least-loaded core
    - Load very balanced load either way. Went with Linux* for maximum portability

Test Scenarios

Test Scenario	Description
run_sarmti_test1	Uniform background with no targets
run_sarmti_test2	Uniformed background with 10 moving targets
run_sarmti_test3	Structured background with no targets
run_sarmti_test4	Structured background with 10 moving targets

Compilers: Gnu vs Intel® C++ Compiler for Linux*

- gcc: Gave better results on FFTW* libraries
  - FFTW libraries developed and optimized using gcc

Compiler Option	Scenario 1	Scenario 2	Scenario 3	Scenario 4
SARMTI code = gcc FFTW code = gcc	7.74 secs	9.9 secs	7.4 secs	13.9 secs
SARMTI code = icc FFTW code = gcc	7.86 secs	10.1 secs	7.5 secs	14.1 secs
SARMTI code = icc FFTW code = icc	10.25 secs	13.4 secs	9.7 secs	18.9 secs
SARMTI code = gcc FFTW code = icc	9.75 secs	13.1 secs	9.3 secs	18.5 secs

- But overall SARMTI had very similar processing times with either compiler

Multi-Threading: Round 2

- 'gprof' rerun – remaining 'hot spots':
  - 54% FFTW Calls
  - 26% Target detection
  - 13% Data compression
  - 2.5% Data Extraction
- FFTW Library replaced by Intel® Math Kernel Library
  - Vector Math functions
  - "fftw wrappers"

	Scenario 1	Scenario 2	Scenario 3	Scenario 4
C vector code time	6.9 secs	9.3 secs	6.9 secs	12.8 secs
MKL Vector Library time	6.4 secs	8.5 secs	6.1 secs	11.8 secs
MKL library speed up	7.2%	8.5%	11.6%	7.8%
MKL FFT Speed up	10.8%	6.1%	6.8%	7.9%
Total MKL Speed up	**18.0%**	**14.7%**	**18.4%**	**15.7%**

Note: Timings are for a 4x 4C "Tigertown" processor system, but MKL speed-up percentages on "Dunnington" and "Nehalem" systems are similar

Threaded Performance Speed-Up: 4x "Tigertown" Processor System

– 16 Physical Cores, 16 HW Threads per system

Test Case	Intel® Xeon® Processor	Serial Time (sec)	Threaded Time 16 Cores	Speed Up 0T→16T
1	4x TGR	85	6.4	13.3X
2	4x TGR	120	8.5	14.1
3	4x TGR	104	6.1	17.0
4	4x TGR	166	11.8	14.1

Serial to 16-Thread Speed Up: 13-17X -- Approaches # of Cores

4x Intel® Xeon® L7345 Processors ("Tigertown") 4 Cores each @ 1.86 GHz each; 8MB L2 Shared Cache; 1 Thread/Core (See configuration information in Backup)

Threaded Performance Speed-Up: 4X "Dunnington" Processor System

-- 24 physical cores, 24 HW Threads

Test Case	Intel® Xeon® Processor	Serial Time (sec)	Time (seconds) per Hardware Cores/Threads						Total Speed Up 0→ 24T	Speed Up 1→24T
			1	2	4	8	16	24		
1	4x DUN	85.4	37.0	18.8	9.9	4.9	3.6	2.2	38.8X	16.8X
2	4x DUN	120.2	47.9	24.6	12.9	6.7	4.8	3.1	38.8X	15.5X
3	4x DUN	104	35.6	18.3	9.6	4.8	3.5	2.1	49.5X	17.0X
4	4x DUN	166.2	67.1	33.9	17.8	9.6	6.6	4.4	37.8X	15.3X

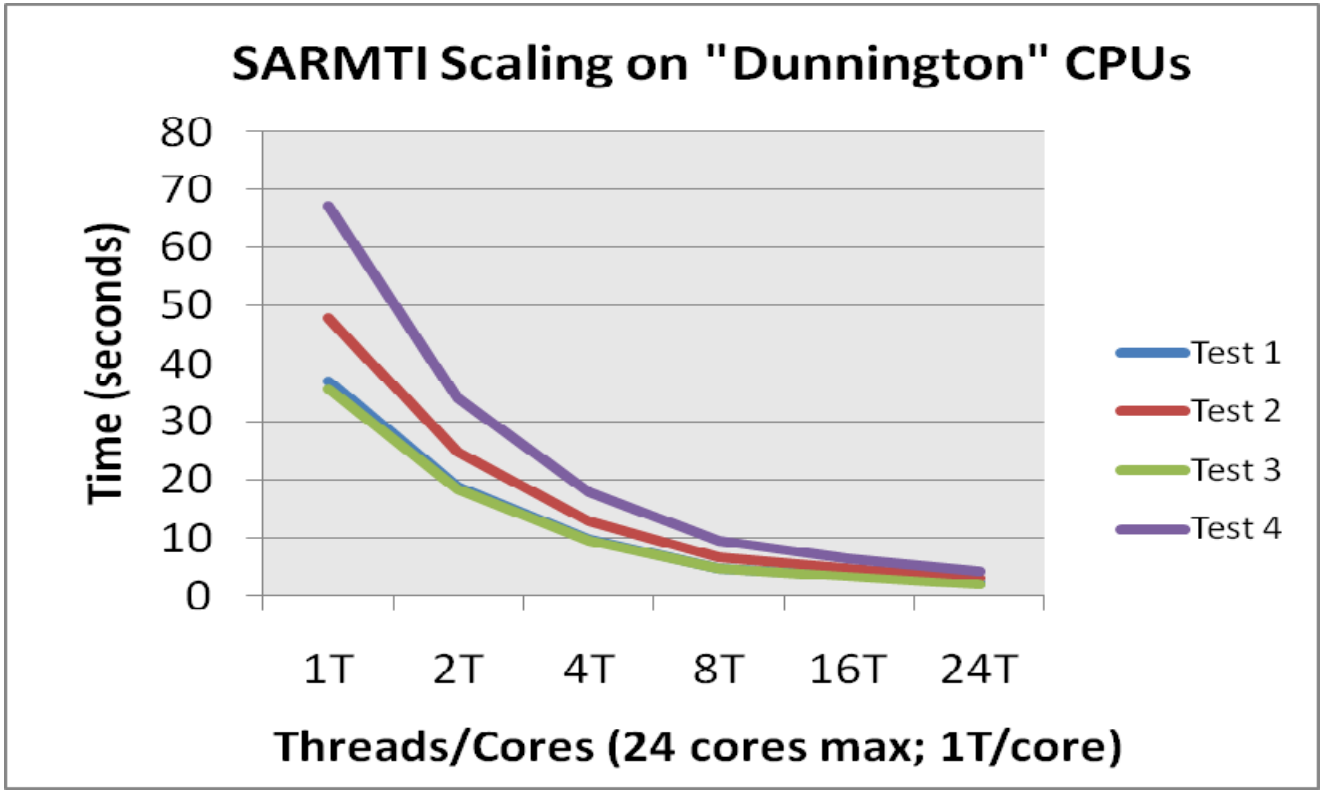
Single Threaded to Multi-Threaded Speed Up: 37-49X

1 Thread to 24 Thread Speed Up: 15-17X

4x Intel® Xeon® MP X7460 Processors ("Dunnington") 6 Cores each @ 2.686 GHz, 16 MB Shared L2 Cache 1 Thread/Core (See configuration information in Backup)

Note that "Dunningtons" are pin-compatible with "Tigertowns"

Performance Scaling / Core: 4x 6 Core "Dunnington"



- Intel processors' large L2 caches are a major factor in the high performance of this application/workload
- Gain % slows > 8 threads, but is still significant overall
  - Different portions of the algorithm use differently sized data sets, whose sizes change dynamically as the geometry changes. Some of the data sets do not thread as efficiently across 16 or 24 cores

Threaded Performance Speed-Up: 2x "Nehalem" Processor System

– 8 Physical Cores, 16 HW Threads

Test Case	Intel® Xeon® Processor	Time (seconds) per Hardware Cores/Threads						Speed Up 1T→8 T	Speed Up 1T→16T
		1	2	4	8	16	24		
1	2x NHL	29.04	15.28	7.87	4.77	4.26	4.19	6.1X	6.8X
2	2x NHL	36.93	19.29	10.04	6.31	5.4	5.61	5.8X	6.8X
3	2x NHL	27.95	14.69	7.57	4.43	4.03	4.16	6.3X	6.9X
4	2x NHL	53.5	27.45	14.15	7.5	7.4	7.73	7.1X	7.2X

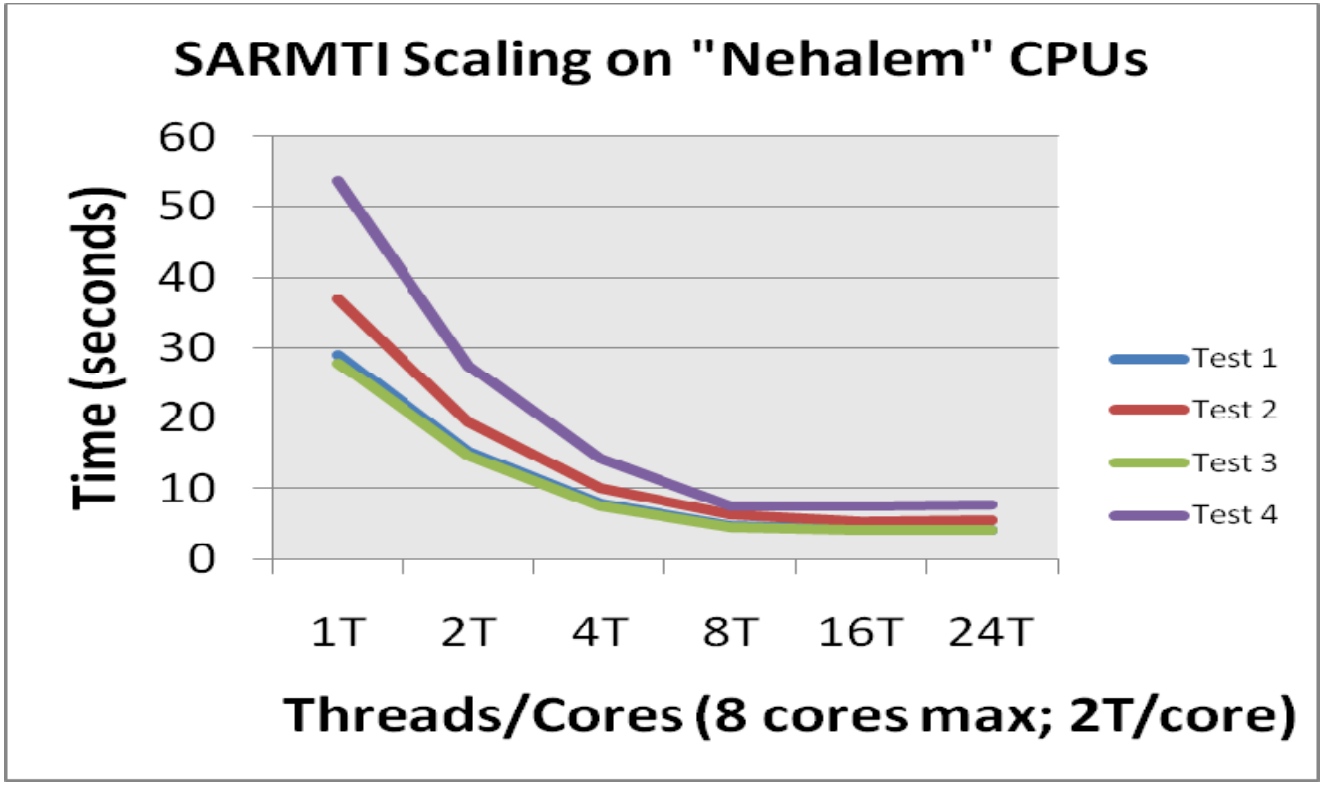
1 Thread to 8 Thread Speed Up: 6-7X

Very Little Gain From Second HW Thread/Core

24 thread time worse because some cores are running > 2 threads

2x Intel® Xeon® DP X5570 Processors ("Nehalem-EP") 4 Cores each @ 2.93 GHz, 2 Threads/Core (See Configuration information in Backup)

Threaded performance speed up: 2x 4 Core “Nehalem”



- Intel processors’ large L3 caches are a major factor in the high performance of this application/workload
- Some of the data sets do not thread efficiently across 16 or 24 cores. Timings for 24 threads are slightly higher because max HW threads on this system is 16

Performance / SWAP Comparison

CPU	Test 4 Time (seconds)			Processor Maximum Thermal Design Power	Deltas					
	8T	16T	24T		Core Speed	8T Time	16T Time	24T Time	Max TDP	System Size
Tigertown		11.8		4X 50 = 200W						
Nehalem	7.5	7.4		2x 95 = 190W	63% faster		63% faster		5% cooler	75% smaller
Dunnington	9.6	6.6	4.4	4x 130 = 520W						
Nehalem	7.5	7.4	7.73	2x 95 = 190W	9% faster	22% faster	11% slower	44% slower	273% cooler	75% smaller
Nehalem	4x				Forthcoming					

4-socket Dunnington system: fastest overall time

Nehalem microarchtecture far more efficient core for core

Phase 2 Study

- Offload FFTs and other parts of SARMTI to FPGA(s)
  - Xilinx* FPGAs/Nallatech* PCIe and In-Socket Accelerator Modules
  - Altera* FPGAs/XtremeData* In-Socket Accelerator Modules
- Demonstrate effectiveness of Intel® QuickAssist Acceleration Abstraction Layer (AAL)

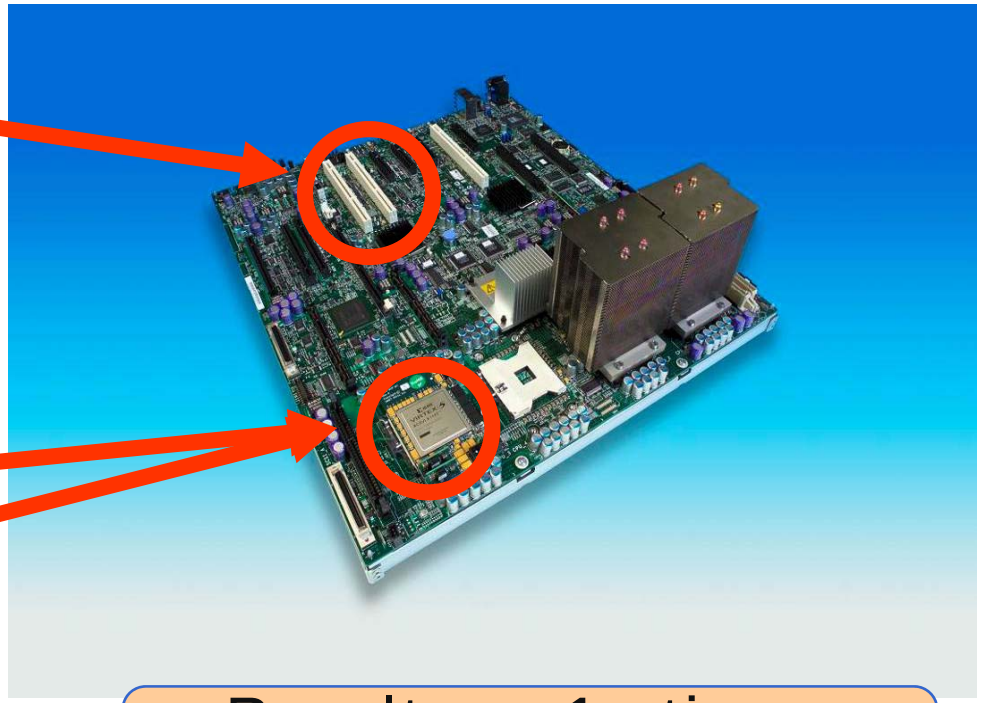
Loosely Coupled FPGA accelerators -- PCIe

NALLATECH

Tightly Coupled FPGA accelerators – Front Side (Processor) Bus

NALLATECH
The High Performance FPGA Solutions Company

XtremeData, Inc.™
COMPUTING REDEFINED

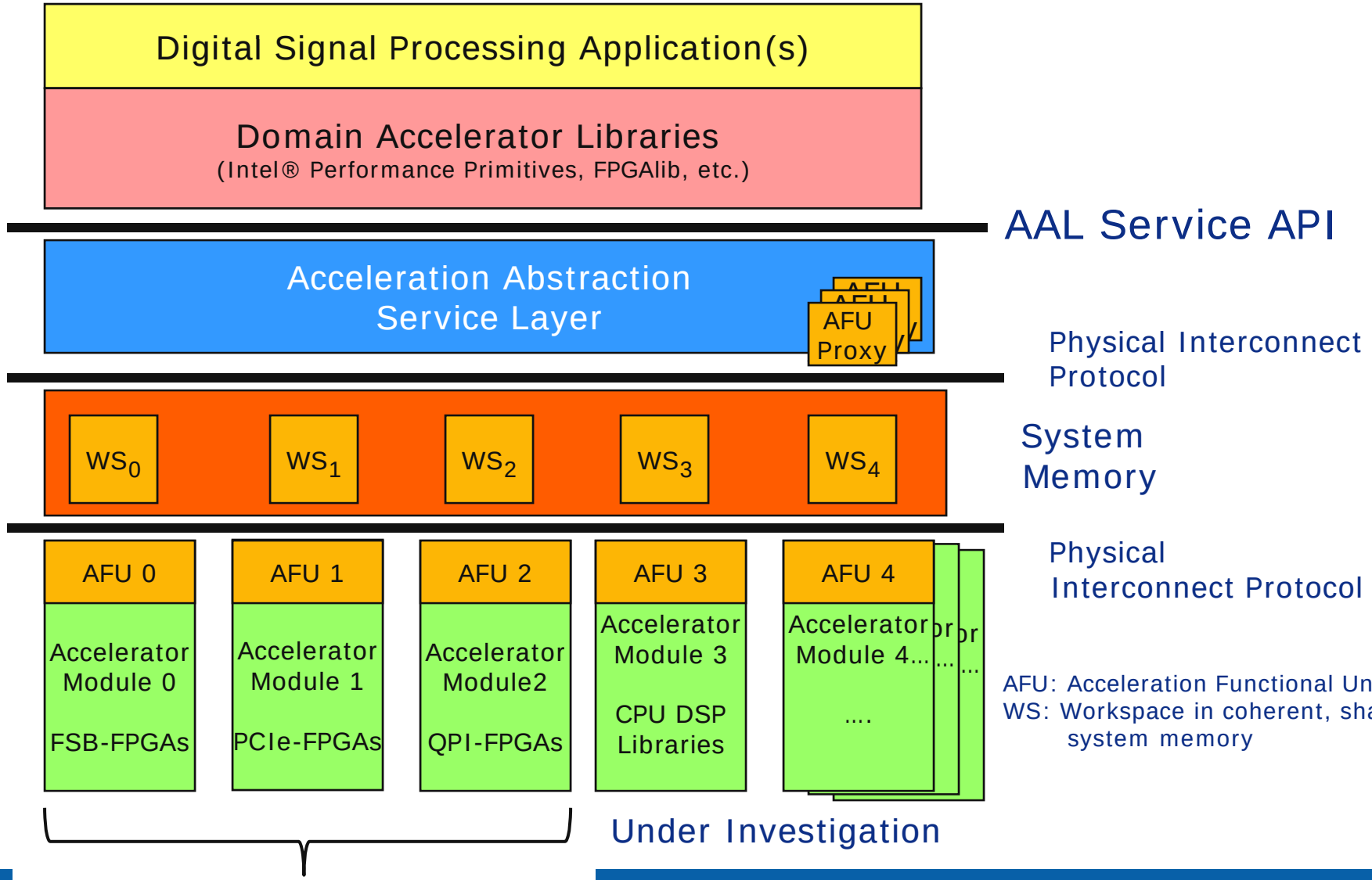


Results: <1s times achievable

Intel® QuickAssist Acceleration Abstraction Layer

- Low level software runtime layer
- Service Oriented Architecture (SOA)
  - Publish/Request service model: database of acceleration functions
  - Virtualizes accelerators: multiple calls operate seamlessly
  - Implements system-wide resource aggregation and sharing policies
- Abstracts hardware details from application
  - Optimizes memory interaction between accelerators and operating system
  - Provides dynamic interface binding to acceleration packages
  - Dynamically reconfigures FPGAs
- Handles all types of accelerated workloads
  - Data parallel and/or task parallel
  - Asynchronous programming model: optimized for concurrent task processing
  - Very large to small data sets; streaming data
- Designed to be compatible with existing software development frameworks

AAL Conceptual Model



Backup

SARMTI POC Configurations

Intel® SFC4UR "Foxcove" 4-processor system -- 4U Rack Mount Server

- Four Intel Xeon® Processors L7345 ("Tigerton") – 4 cores each @ 1.86 GHz; 8MB L2 cache 50W Max Thermal Design Power each
- Four Intel Xeon Processors MP X7460, 6 cores each @ 2.66 GHz, ("Dunnington"); 16MB L3 cache. 130W Max Thermal Design Power
- 1066 MHz Front Side Bus
- 16 MB 667-MHz FBDIMMs
- Fedora* release 8 (Werewolf*) for x86_64 architecture (Linux* 2.6.23 kernel)
- gcc 4.1.2, flags: -O3 –Xt –ip –fno_alias –fargument-noalias
- Intel Math Kernel Library version 10.0

Intel® SR1625 "Petrof Bay 2-processor server – 1U Rack Mount Server

- 2 x Intel® Xeon Processors X5570 ("Nehalem-EP") 4 cores each @ 2.93 GHz; QPI: 6.4 G transfers per second, 8M L2 Cache per processor 95W Max Thermal Design Power per processor
- 12 MB 1333MHz Registered ECC DDR3
- SCSI HARDDRIVE Seagate 146GB Savio 10k RPM SAS 2.5in SFF
- NUMA Optimized [Enabled] in BIOS
- Frequency scaling off: **/sbin/service cpuspeed stop**
- Centos 5.3 el5_x86_64 (RHEL) (Linux* 2.6.18 kernel)
- gcc 4.1.2, flags: -O3 –Xt –ip –fno_alias –fargument-noalias
- Intel Math Kernel Library version 10.2
- Intel Integrated Performance Primitives 6.1

Intel® QuickAssist Technology Accelerator Abstraction Layer White Paper:

http://download.intel.com/technology/platforms/quickassist/quickassist_aal_whitepaper.pdf