Clearspeed Embedded Apps and Architecture for Space

EEL 6686: Presentation 1

Chris Morales Kaz Onishi

ECE
University of Florida, Gainesville, Florida

January 29, 2015

Introduction

- Embedded systems for space require high performance/watt
  - Limited resources in space
  - Applications are becoming more complex
- Example applications include:
  - Signal Processing
  - Controls
  - Real time system applications
- Hardware must meet design requirements for space missions
  - Reliability concerns
  - Power constraints
  - Perform task within reasonable time

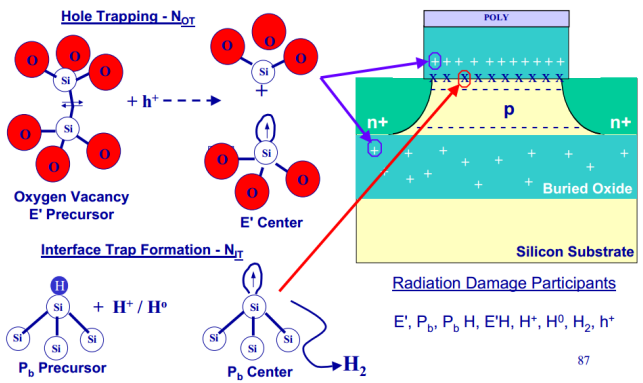
Reliability

- Why is reliability an issue in space?
  - Components are expensive to bring up, don't want an expensive failure
  - Operational for several years
  - Safety Critical missions
  - Cannot repair broken parts easily
- Challenges for reliability
  - Human errors
  - Degradation
  - Mechanical stresses
  - Radiation

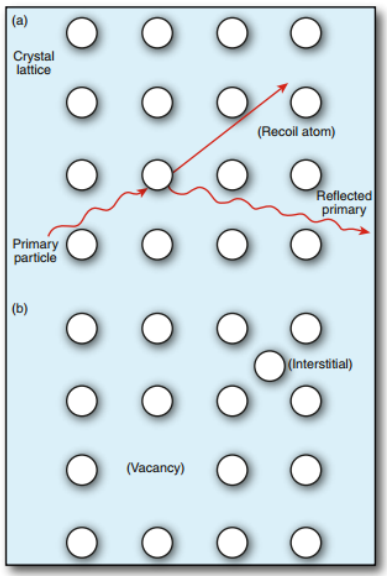
Radiation Effects in Space (1/2)

Two types of Permanent Radiation damage

- Total Ionizing Dose(TID)
  - Changes circuit level device behavior
  - Changes threshold voltage leading to timing issues
- Displacement
  - Radiation particles scatter off lattice deforming structure
  - Damage dependent on energy of particle



(a) TID [2]



(b) Disp [4]

Radiation Effects in Space (2/2)

Several types of Non-Permanent damage

Single Event Effects:

- Single Event Transient (SET)
  - Voltage of part becomes incorrect due to charged particle
  - Circuit outputting wrong voltage for brief time
- Single Event Upset (SEU)
  - SET can lead to SEU if data going to memory is corrupt
  - Radiation hitting memory causing bit flip
  - Can go unnoticed if data is not being used
- Single Event Functional Interrupt (SEFI)
  - Happens when radiation hits critical point
  - Hard to fix, usually requires a full system reset

Solutions for Reliability

Radiation Hardened (RadHard) components

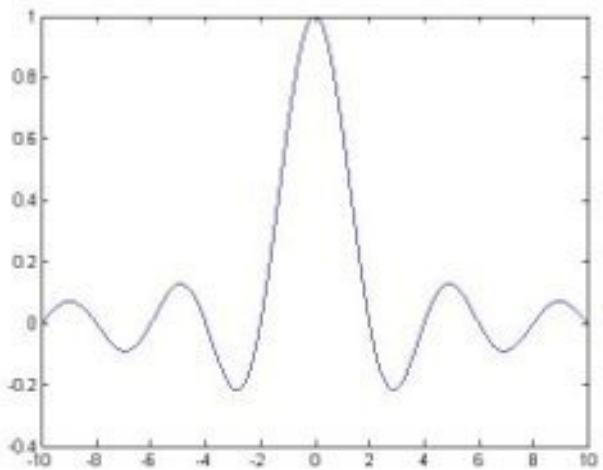
- Expensive Process
- Lengthy process that takes years
  - RadHard processors lag behind industry standards
  - Usually lags behind 3-5 years
- Strong against radiation damage such as TID and displacement

Redundancy

- Have extra information/hardware
  - Have a voter system
  - Unlikely to have same error on different devices
- Big overhead leading to performance loss
- Simple implementation (compared to RadHard)

Radiation Hardened RADSPEED DSP

- Designed as an accelerator to host processor
- Multi-core DSP
  - Optimize for signal processing and high performance
  - Power constraints must be met
- Applications that run well with architecture:
  - RF processing apps
  - Radar processing apps
  - Hyper-spectral imaging
  - Image processing



RADSPEED DSP Architecture (1/4)

- Includes 160 Processing Elements (PEs)
  - Arrayed in two groups of 76 elements
  - Includes 8 spare processors
- Each array is a multi-threaded array processor (MTAP)
  - Controlled by mono processor
  - Each PE includes a double precision floating point core
  - Includes a small low powered integer processor

RADSPEED DSP Architecture (2/4)

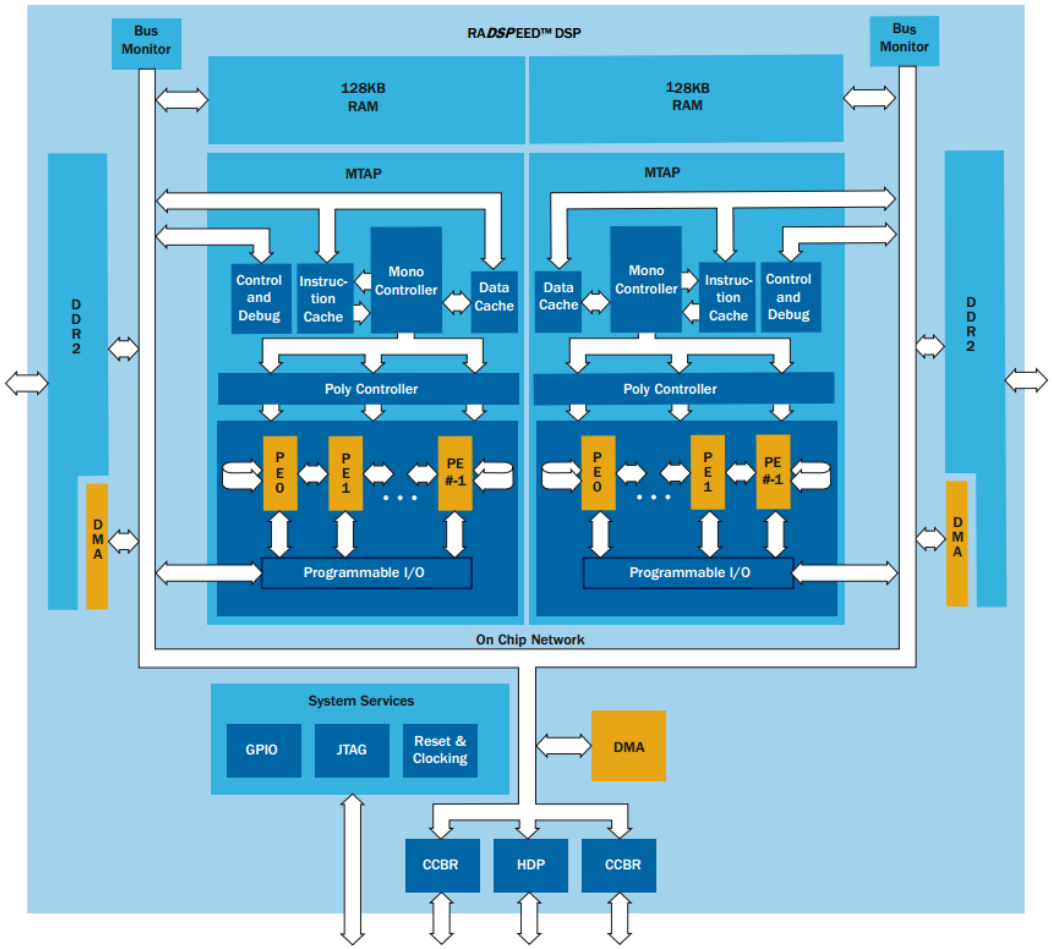
- Each MTAP acts as an independent core
  - Each RADPSEED card comes with 2 MTAPs thus 2 cores
  - Each MTAP operates as SIMD architecture
- Each array of PEs includes a 30 Gbps dedicated DMA controller
  - This DMA is used for DDR2 memory interface
  - Fast enough to avoid bottleneck during data transfer

RADSPEED DSP Architecture (3/4)

- DSP includes two 30 Gbps ClearConnect Bridges (CCBR)
  - Not the same as the DMA controller
  - Provides external data interface to MTAPs
  - Provides DDR like interface to MTAPs
- Dual CCBR is a scalability improvement over CSX700
  - Allows for daisy-chaining multiple RADSPEEDs
  - Supports up to four RADSPEEDs

RADSPEED DSP Architecture (4/4)

RADSPEED DSP Architecture



CSX700 vs RADSPEED

- RADSPEED is a variant of the Clearspped CSX700 DSP
  - RadHard technology with same architecture
  - Number of PEs reduced from 192 to 152
  - Clock speed reduced from 250Mhz to 233 Mhz
  - Peak performance reduced from 96 GFLOPs to 70 GFLOPs
- Minimal change makes CSX700 great candidate for space processor
  - Less impact on performance
  - Utilizes parallization instead of clock speed

Why CSX700?

Comparison between different RadHard technology to their counterparts

	RADSPEED	CSX700
Clock	233MHz	250MHz
Processing Elements	152	192
Mono memory	128KB	128KB
Poly memory	6KB	6KB

(c) RADSPEED vs CSX700

PowerPC e5500	P5040DS	RAD5545
Clock	1.8 - 2.4 GHz	1GHz
2 L1 cache (Instr or Data)	32KB	?
L2 cache backside	512KB	?
L3 cache frontside	2048KB	?

(d) Rad5545 vs P5040

Other On-Board Processing Hardware Choices

Application Specific Integrated Circuit (ASIC)

- High performance/watt
- Limited functional flexibility for embedded systems

Field Programmable Gate Array (FPGA)

- Less performance than ASIC but a little more flexible
- More susceptible to radiation
  - Configuration setting are crucial usually stored in SRAM
  - Anti-fuse and flash-based FPGA are better but lack re-configurablility

Graphics Processing Unit (GPU)

- Very high performance
- Very high power requirements

Why is Optimizing important?

- Moore's Law is reaching an end
  - Can no longer increase clock speed (frequency wall)
  - Device scaling is reaching a limit
- How can we achieve higher performance now?
  - Parallelization increasing in demand for better performance
  - Efficient use of architecture
  - Architectures are utilizing more cores for higher parallelism
- Future will be more in optimizing in software than hardware

Optimization on Parallel Architectures (1/2)

Amdahl's Law

$$Speedup = \frac{1}{P + \frac{1}{n}(1 - P)}$$

Where:
P: Fraction of algorithm that is strictly serial
n: Number of parallel units

- Amdahl's Law is stating diminishing returns
  - Need to parallelize majority of program to get good speedup
- Efficient use of PEs
  - Want all PEs to be doing computation at all times
  - PEs that are waiting on data or communicating wastes time

Optimization on Parallel Architectures (2/2)

- Efficient external memory communication
  - Reported 80% of total time in communications for CSX700
  - Overlap communication with computation to reduce overhead
  - Transfer data in blocks equivalent to the buffer size
- PE memory size
  - Dedicated memory for each PE is limited
  - Communication overhead for external memory to Poly memory
- Vector Operations
  - Sequential operations take time due to communication overhead
  - Vector instructions operate on multiple data must be used

Introduction to CAF algorithm (1/2)

The Cross/Complex Ambiguity Function (CAF)

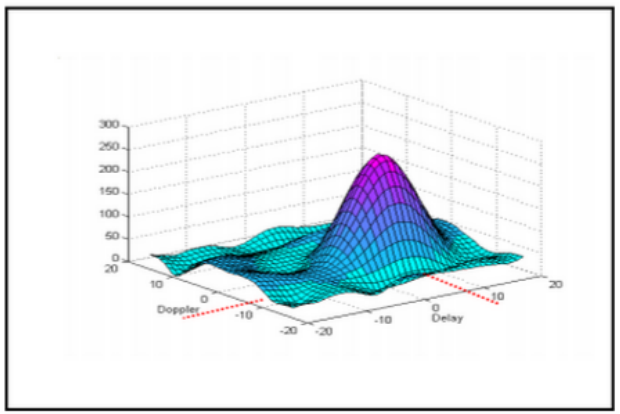
$$X(\tau, v) = \int_{-\infty}^{\infty} s_1(t) s_2^*(t+\tau) e^{j2\pi vt} dt$$

Where:
τ: Time delay
v: Frequency offset or Doppler

- CAF is two dimensional function
- Maps 2D planes of the time distance of arrival(TDOA) and frequency distance of arrival(FDOA)

Introduction to CAF algorithm (2/2)

- Peak energy provides true delay and doppler of correlated signals
- CAF on RADSPEED
  - First merge two complex data streams into 3d array
  - Second perform FFT on results of the merged function
  - Take absolute value of FFT 3D array to get real values
  - Sum time difference steps to create 2D array



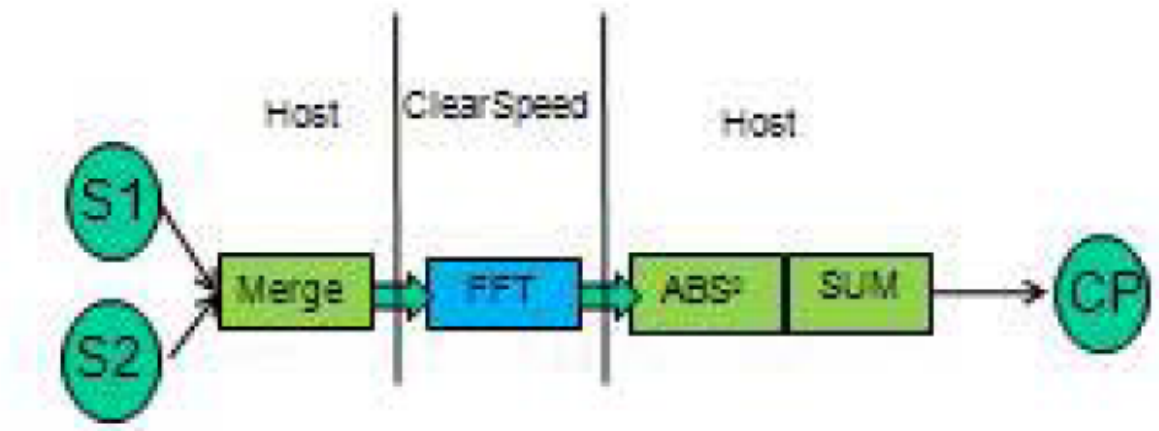
Output CAF Surface

Experimental Setup

- 2.8Ghz Intel core as host
  - Used this as the baseline performance test
  - Baseline involves running entire app on this device serially
- CSX700 was used as an accelerator to the Intel chip
- Optimization steps
  - Different steps of optimization were compared to the baseline
  - More optimizations = more calculations done on CSX700
- RADSPEED data approximated from CSX700 results

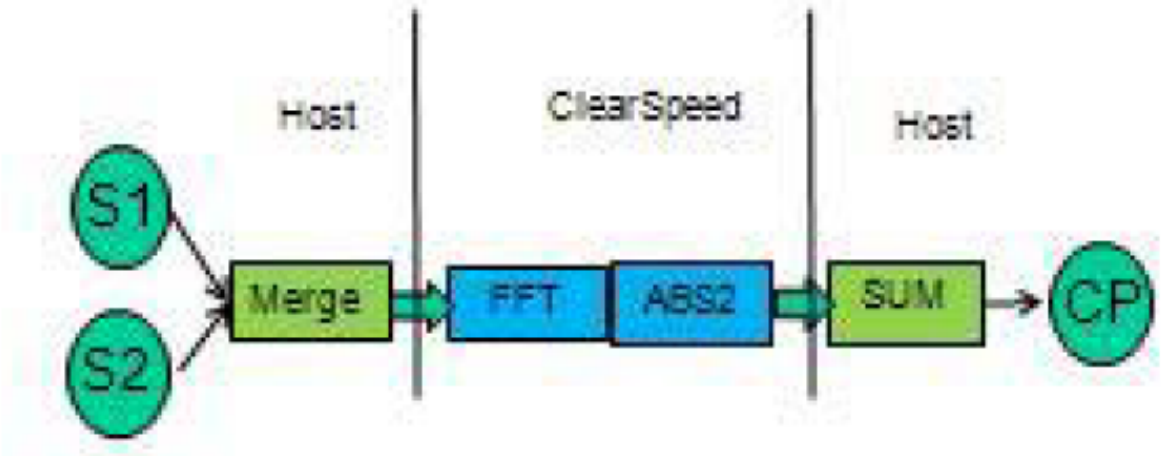
Optimizations on CSX700 for CAF (1/5)

- Perform FFT on DSP and utilize the SIMD MTAPs
- FFT runs faster than host
  - Memory bottleneck occurs
- Optimization reduced by 3D array output



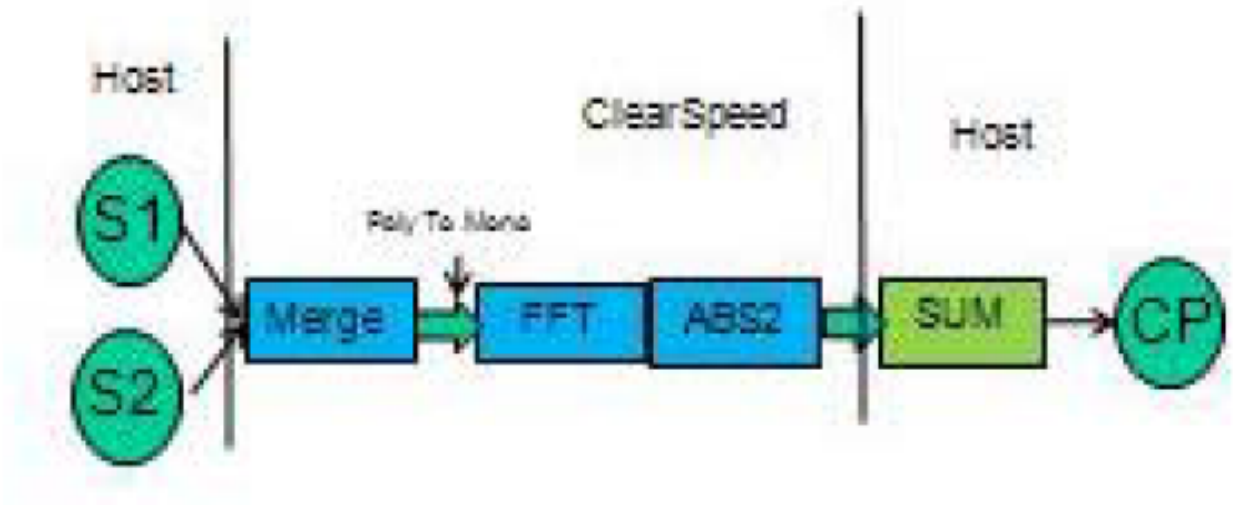
Optimizations on CSX700 for CAF (2/5)

- Perform abs function on DSP to reduce memory transfers
  - abs^2 function can be done before memory transfers
- This reduced transfer by 2 times



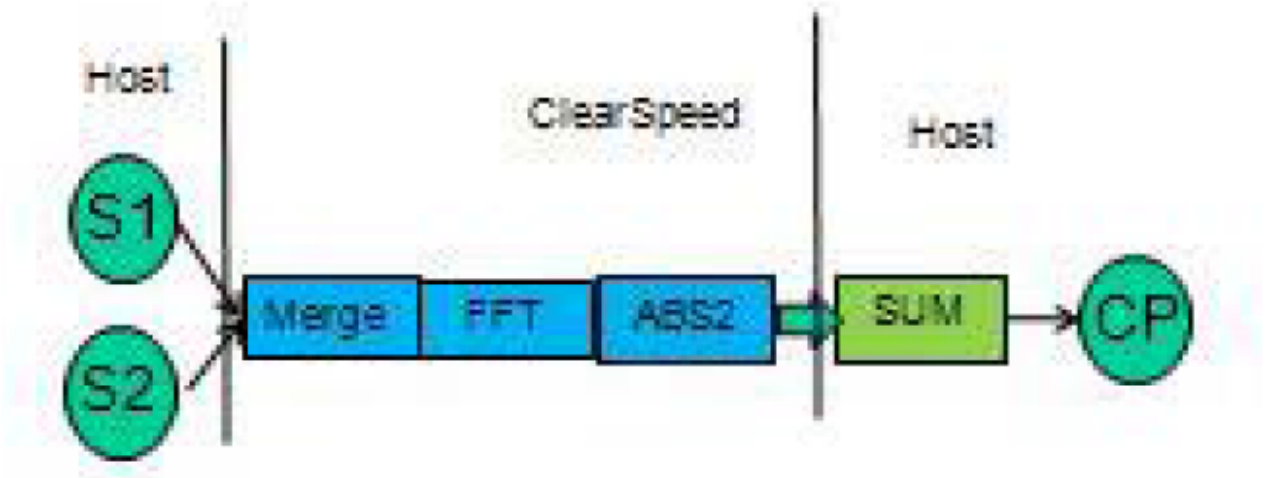
Optimizations on CSX700 for CAF (3/5)

- Put merge function on DSP
- Reduces transfers from host
  - Reduces from size*steps*blocks to (size + steps)*blocks



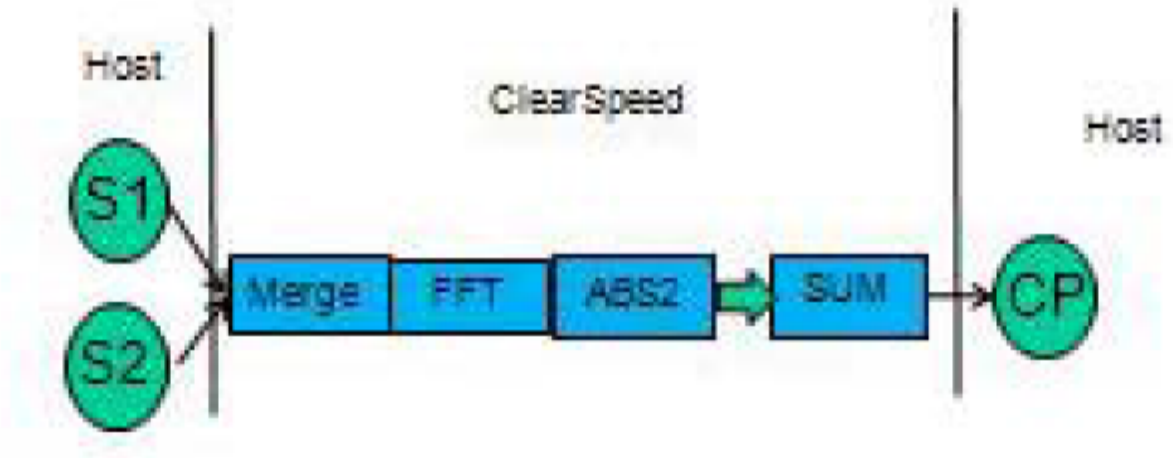
Optimizations on CSX700 for CAF (4/5)

- Eliminate transfer of merge output from poly to mono
- Have a master function which performs all 3 tasks



Optimizations on CSX700 for CAF (5/5)

- Last perform sum function on DSP
- Reduces transfers to host
  - Reduces from size*steps*blocks to size*steps



Results of Different Optimization Stages

- From baseline more than 10x performance was achieved
- RADSPEED results scaled from CSX700

Optimization	iterations/sec 512x1024x72	Improvement vs. baseline	FLOPs/SEC
FFTW (1x2.8Ghz Intel Core)(baseline)	0.358	(base line)	743,272,611.84
Opt 1 (FFTs on CSX600)	0.400	1.1	830,472,192.00
Opt 2 (FFTs and ABS on CSX600)	0.591	1.7	1,227,022,663.68
Opt3 (Merge <-> FFT, ABS on CSX600)	0.929	2.6	1,928,771,665.92
Opt4 (Merge FFT, ABS on CSX600)	1.089	3.0	2,260,960,542.72
Opt5 (Merge FFT, ABS->SUM on CSX600)	2.053	5.7	4,262,398,525.44
Opt5 CSX700 vxWorks(192pes 250Mhz)	4.940	13.8	10,256,331,571.20
Opt5 RADSPEED (152 pes 233Mhz) (estimate	3.645	10.2	7,567,463,310.95

Other Applications on the CSX700 Architecture

- 4 algorithms were test on CSX700 in paper
  - Harris Corner Detector (HCD)
  - Stereo Vision
  - Random Sample Consensus (RanSaC)
  - Histogram of Oriented Gradient (HOG)
- Go over results of 2 algorithms that sums up performance
  - HCD and HOG
  - other algorithms showed similar results

Harris Corner Detector

Comparisons against GPU, FPGA, ASIC

- Performs better in terms of latency and fps
- Provides a high degree of flexibility for size

Image Resolution	Latency (ms)		fps		Sustained GFLOPS	
	$HCD_{3\times3}$	$HCD_{5\times5}$	$HCD_{3\times3}$	$HCD_{5\times5}$	$HCD_{3\times3}$	$HCD_{5\times5}$
128x128	.165	.224	6060	4464	3.97	4.68
352x288	.8	1.22	1250	819	5.06	5.31
512x512	1.74	2.63	574	380	6.02	6.37
640x480	2.15	3.28	465	304	5.71	5.99
1280x720	7.04	10.89	142	91	5.23	5.41

Image Resolution	fps reported in [ref]	fps achieved by our approach
128x128	1367 (ASIC) [13]	4464
352x288	60 (FPGA) [14]	819
640x480	99 (GPU) [15]	304

Histogram of Oriented Gradient

Performed worse than GPU

- Has significantly higher fps/watt
- Goal for embedded systems in space is lower power

	Latency (ms)	Peak Performance (GFLOPS)	fps / watt
CSX [41]	146.23	96	.75
GPU [39]	99	384	.06
GPU [40]	67	1788.48	.05

Conclusion

- Optimization plays a huge role in performance
  - Especially in parallel architectures
  - If not optimized correctly performs worse than serial
  - Optimizing correctly gets massive performance boost from the same hardware
- CSX700 is excellent architecture for low power embedded apps
  - Utilizes huge amounts of parallelism
  - Very efficient performance/watt

Future Work

- CAF is only one algorithm
- More benchmarks must be performed to understand full potential
- Additional algorithms for space involve:
  - Matrix multiplication
  - Data compression
  - Integer sort
  - Anomaly Detection
- The comparisons between CSX and other architectures did no look fair

References

Fijany, A., and Hosseini, F.

Image processing applications on a low power highly parallel simd architecture.

In *Aerospace Conference, 2011 IEEE* (March 2011), pp. 1–12.

Kayali, S.

Space radiation effects on microelectronics.

Tech. rep., NASA Radiation Effects Group.

Marshall, J., Berger, R., Bear, M., Hollinden, L., Robertson, J., and Rickard, D.

Applying a high performance tiled rad-hard digital signal processor to spaceborne applications.

In *Aerospace Conference, 2012 IEEE* (March 2012), pp. 1–10.

Maurer, R. H., Fraeman, M. E., Martin, M. N., and Roth, D. R.

Harsh environments:space radiation environment, effects, and mitigation.

Tech. rep., Johns Hopkins APL Digest, 2008.