Jet Propulsion Laboratory
California Institute of Technology

Flight Validating Artificial Intelligence Software on the Qualcomm Snapdragon Processor Onboard the ISS

Faiz Mirza

ISS R+D Conference, August 16th, 2021

Team

- Jet Propulsion Laboratory, California Institute of Technology
  - Faiz Mirza
  - Jason Swope
  - Dr. Emily Dunkel
  - Dr. Zaid Towfic
  - Dr. Damon Russell
  - Dr. Joseph Sauvageau
  - Dr. Douglas Sheldon
  - Dr. Steve Chien
- Hewlett Packard Enterprise
  - Dr. Mark Fernandez
  - Carrie Knox

Introduction

- Flying autonomy software onboard spacecraft can increase the autonomy
  - Increase science return and decrease dependency on human operators.
  - Requires more computing resources than traditional spacecraft flight software: CPU throughput and memory.
- We benchmark several autonomy and instrument processing algorithms on the Qualcomm Snapdragon 855, a "system on a chip".
- The primary metric is the runtime of the sample scenarios.
- Secondary metric is power consumption
- We run onboard Snapdragon 855 connected to SBC2 on the ISS

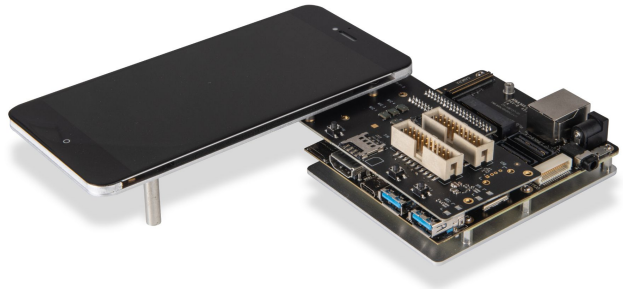
Summary

- General Algorithms
  - Fast Fourier Transform Benchmarks
  - Matrix Multiplication Benchmarks
- Planning Algorithms
  - MEXEC (Multi-Mission Executive)
  - CLASP (Compressed Large-Scale Activity Scheduling and Planning)
  - Copilot (M2020 Ground Scheduler)
- Instrument Processing
  - Synthetic Aperture Radar (SAR) Image Formation
  - Hyperspectral Compression
  - High-Order Wavefront Sensing
  - Match Filters
- Machine Learning
  - Flood mapping with SAR Imagery
  - Mars Reconnaissance Orbiter (MRO) landmark classification
  - MSL Mars Rover image classification
  - MSL image classification for health analysis
  - Salience for landmark detection
  - Storm Classification

Autonomy FSW on Conventional Flight CPU's

- AI-based Planners have flown onboard spacecraft as early as 1999
- Remote Agent Experiment onboard Deep Space 1 flew for 48 hours *(Jonsson et al. 2000)*
  - Demonstrated closed-loop, goal-based commanding onboard a RAD 6000 PPC
- The Autonomous Sciencecraft Experiment flew CASPER on Earth Observing 1 for over 12 years (2003 - 2017) *(Chien et al. 2005)*
  - Encountered limited observability, limited memory, and limited CPU constraints onboard a Mongoose V
- Intelligent Payload Experiment in 2013 ran CASPER again as well some machine learning applications *(Chien et al. 2016)*
  - Used an Atmel ARM9 chip running Linux
- ASTERIA was a 6U Cubesat running Linux on a Cortex 160 *(Troesch et al. 2020)*
  - Ran several MEXEC scenarios, encountered similar memory and cpu limitations
- Support Vector Machines and Random Decision Forests have been run on EO-1 on a Mongoose V (2005 - 2017?)
- Deep Learning Models were run onboard Φ-Sat using a Intel Movidius Myriad II (2020?)

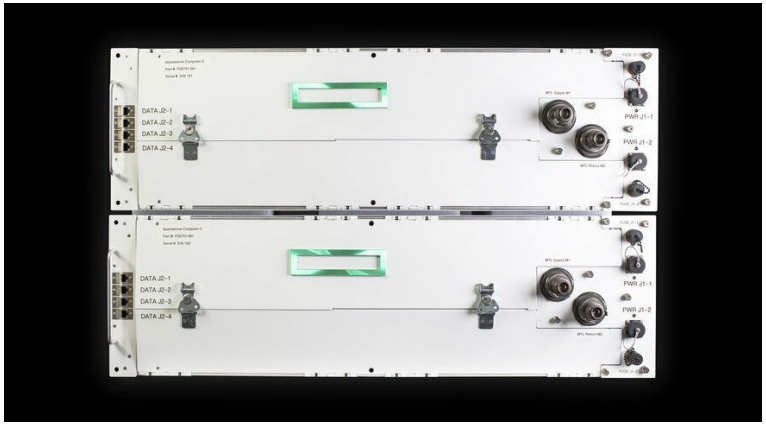
Qualcomm Snapdragon 855



- 8 core ARM system
  - 4 “silver” high efficiency cores ~ 1.80 GHz
  - 3 “gold” high performance cores ~ 2.42 GHz
  - 1 “gold prime” very high performance core ~ 2.84 GHz
- Adreno 640 GPU
- Qualcomm Hexagon 690 Digital Signal Processor
- Neural Processing Engine
- Running Android OS

Spaceborne Computing 2 (SBC2)

- Commercial, off-the-shelf linux workstations from Hewlett Packard Enterprise (HPE)
- Intel Xeon 5215 Processor with 10 cores
  - Each core can run 2 threads simultaneously
- 4 NVIDIA Tesla GPUs
- 2 Machines aboard the ISS;
  2 Machines on the ground (testbed)
- Launched to the ISS 21 February 2021
- The planning applications use SBC2 as the comparison baseline



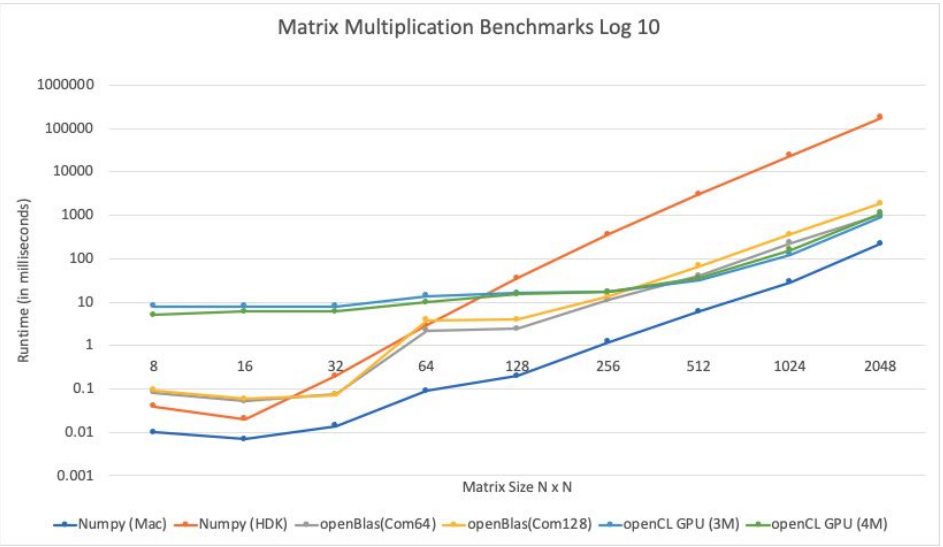
Porting to Android

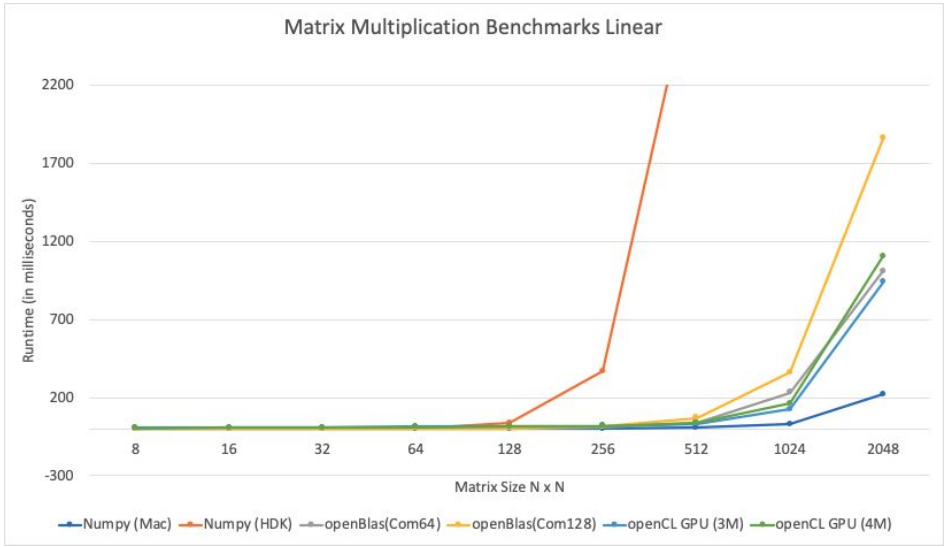
- Most applications involve cross-compiling C/C++ code for ARMv8-A architecture
- ARM only ports are fairly straightforward
  - Some single threaded, some multithreaded
- Some applications are ported to GPU, DSP and/or NPU as well
- Some applications in python, ported using the python-for-android library provided by Kivy
- Eventual goal is to benchmark all applications on all processors that make sense for that application

Fast Fourier Transform (FFT)

Duration Power Current	Float64 ARM	Float32 ARM	Float32 GPU	UINT8 ARM (FASTCV)	UINT8 DSP (QHL)	UINT8 DSP (QCOM)
	← 2X →	← 5X →	≈	← 1.75X →	← 2.75X →	
1D FFT (1x2048)	**0.032ms** 2.5W 600mA	**0.025 ms** 2.5W 600mA	**0.025 ms** 2.1 W 525mA	**0.025 ms** 2.5W 600mA	**0.774 ms (1 thread)** 1.50W 420mA	
2D FFT (2048x2048)	**928 ms** 2.4W 550mA	**492 ms** 2.5W 600mA	**105 ms** 2.3 W 575mA	106 ms 2.5W 600mA	125.07 ms (1 thread) 2.00W 500mA 79.58 ms (2 thread) 2.47W 600mA **60.69 ms (4 thread)** 2.76W 670mA	
2D FFT (1024x1024)	**78.11 ms** 3.06W 770mA	**63.18 ms** 3.00W 766mA	**23.436 ms** 2.35W 590mA 20% GPU Utilization	**23.56 ms** 3.00W 766mA	26.33 ms (1 thread) 2.0W 500mA 17.00 ms (2 thread) 2.32W 600mA **14.14 ms (4 thread)** 2.60W 640mA	**5.128 ms (4 thread)** 2.40W 612mA

Matrix Multiplication





Generic benchmarks: more extensive parameter changes relating to different libraries and input data sizes.

MEXEC

- Separately threaded Planner and Controller
- Takes “Task Network” as input
  - Set of state timelines, task templates, and tasks
- Generates conflict free plans and monitors task execution, responding to deviations or exogenous events

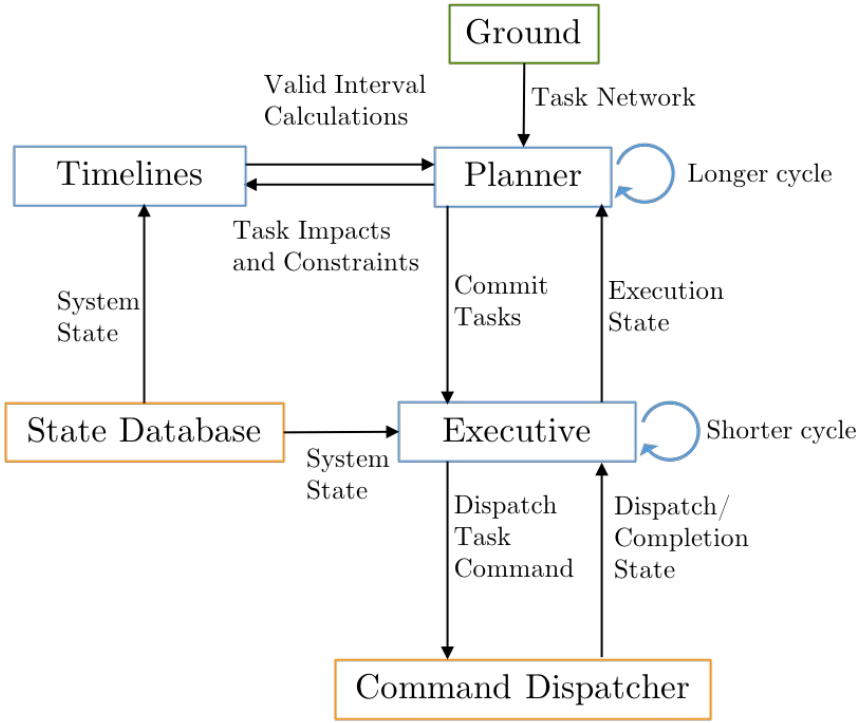
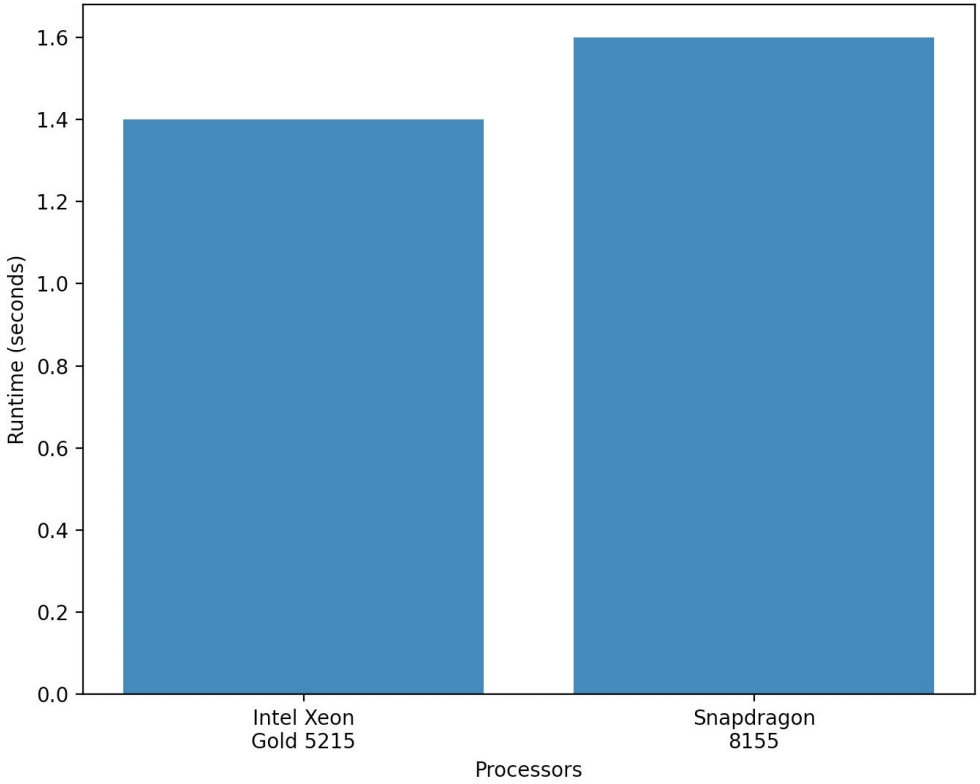


Image from https://ai.jpl.nasa.gov/public/projects/mexec/

MEXEC Benchmark Scenario

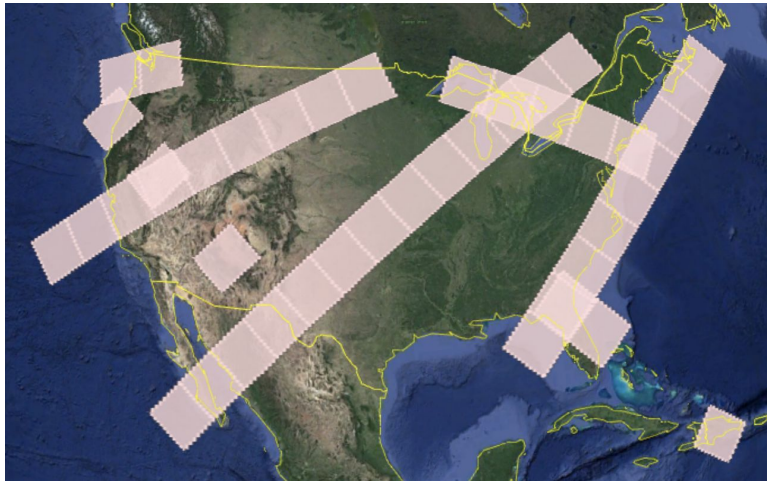
- MEXEC consists of multiple components, but the most computational demanding is the planner, so that is used for benchmarking purposes.
- MEXEC also runs continuously on a cycle, for benchmark purposes we only time the first plan generation.
- As a benchmark, we use the Europa Lander Prototype test scenario *(Wang et al. 2020)*
- Multi-day schedule, exercises hierarchical planning, valid interval search, constraint satisfaction, etc.

MEXEC Results



CLASP

- CLASP is the Compressed Large-scale Activity Scheduler and Planner
- Spacecraft, instrument, and orbiting body models define the scenario
- Science Campaigns define the scientific goals
- CLASP generates an observation schedule based on the scenario constraints

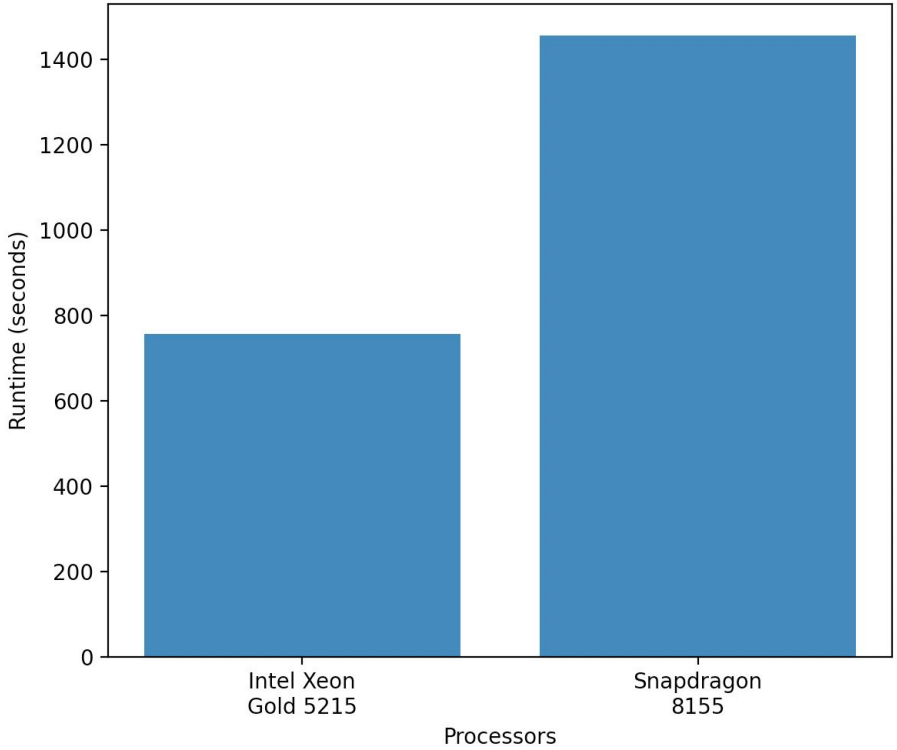


ECOSTRESS schedule portion of the Contiguous United States

CLASP Benchmark Scenario

- CLASP has been used for NISAR, ECOSTRESS, EMIT, OCO3, and others.
- As a benchmark, we generate 2 years of 2 week schedules using ECOSTRESS data from 2018-2020
- For faster timing metric collection, we time generation of a single 2 week schedule
- Currently CLASP just runs single threaded
  - CLASP performs a large amount of ray tracing to compute feasible observations (target visibility) which would benefit greatly from GPU acceleration

CLASP Results



Copilot

- M2020 ground scheduler for scheduling wake/sleep and preheats in operations
- Uses the same scheduling algorithms as the M2020 onboard scheduler
- Challenges include wake/sleep constraints, preheat constraints, variability in execution, and complex operations handover handling.

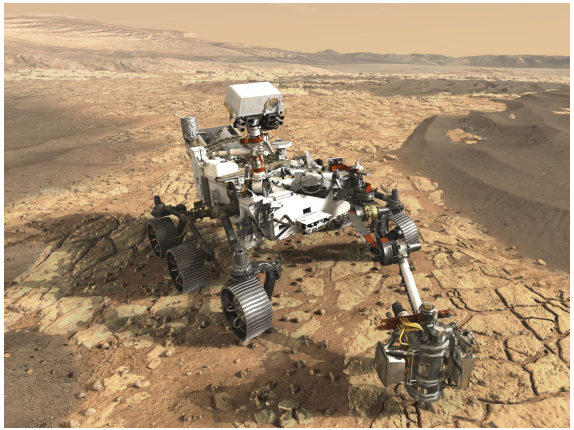
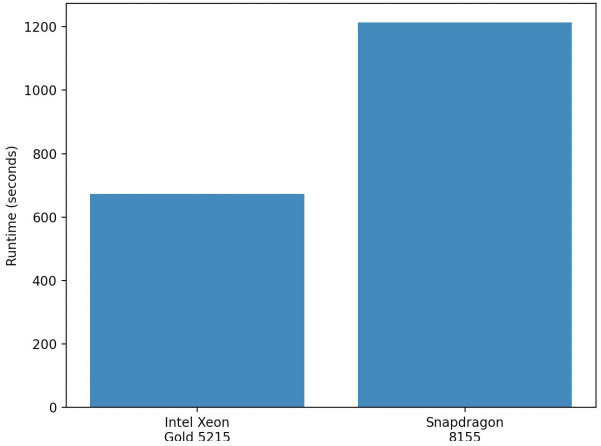


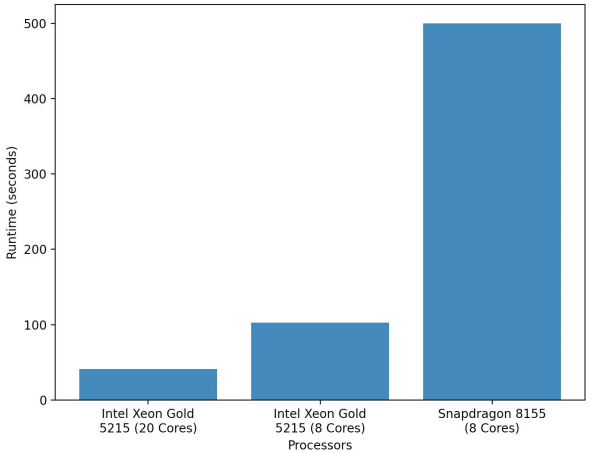
Image from https://ai.jpl.nasa.gov/public/projects/m2020-scheduler/

Copilot Benchmark Scenario

- For this benchmark, we are running with ~800 x 1 martian day (or sol) planning problems that are generated by random variation of 7 base plans or “sol types”
  - Vary execution durations, incoming/outgoing energy state, and alternative action options
- First surrogate port ran each sol type serially, single threaded.
- Second pass parallelized across all cores
  - Large problem already split into 800+ small problems, so easy to parallelize
- Benchmarked against SBC2 using 1 core, 8 cores (to match 855) and 20 cores

Copilot Results

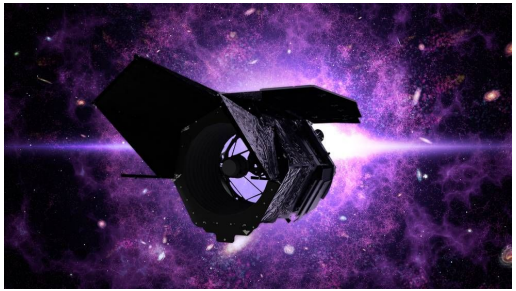




- Runtime of processors on the Copilot benchmark problem
  - Top: Serial
  - Bottom: Parallelized
- On the Intel, all 10 cores are equally good, on the Snapdragon, 1 core is faster than the others

High Order WaveFront Sensing (HOWFS)

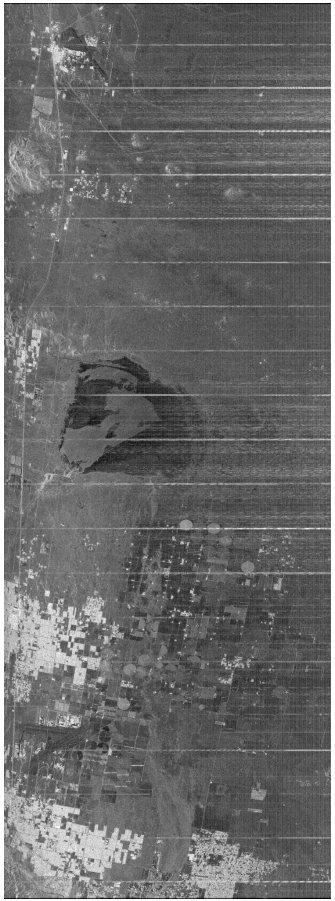
- Python implementation of HOWFS for Roman Space Telescope Coronagraph Instrument
- Using wide field of view test data
- Two tiers of goal
  - Slower mission goal is 7.6 hours
  - Faster mission goal is 30 minutes
- Single-threaded ARM port only
  - GPU possible, some extra work required to work with python
  - Multithreaded approach would require code to be ported or Linux OS to be used
    - python multiprocessing library unsupported on Android
- Original double precision code took 2.2 hours
- Moving to single precision got us to 1.8 hours



https://roman.gsfc.nasa.gov/

Synthetic Aperture Radar (SAR) Image Formation

- Image Processing pipeline from Uninhabited Aerial Vehicle SAR (UAVSAR) instrument
- A pipeline of 3 ARM applications, 2 GPU applications
  - Mainly a row-wise and column-wise 2D FFT with filters applied
- Goal of <240 Seconds
- Currently takes 220 seconds
  - Could possibly be further improved, as GPU usage is only at about 60%



The Rosamond Calibration Array (RCRA), located near the south beach of Rosamond Dry Lake Bed in California.

Hyperspectral Compression

- Benchmarked on test images for Earth Surface Mineral Dust Source Investigation (EMIT)
- 64 lines, 640 samples per line, 481 spectral bands
- ARM only port, GPU port in progress
- MSamples/sec = lines * samples per line * bands / runtime
- EMIT Target is 23.1 MSamples/sec

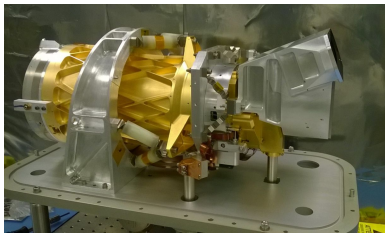


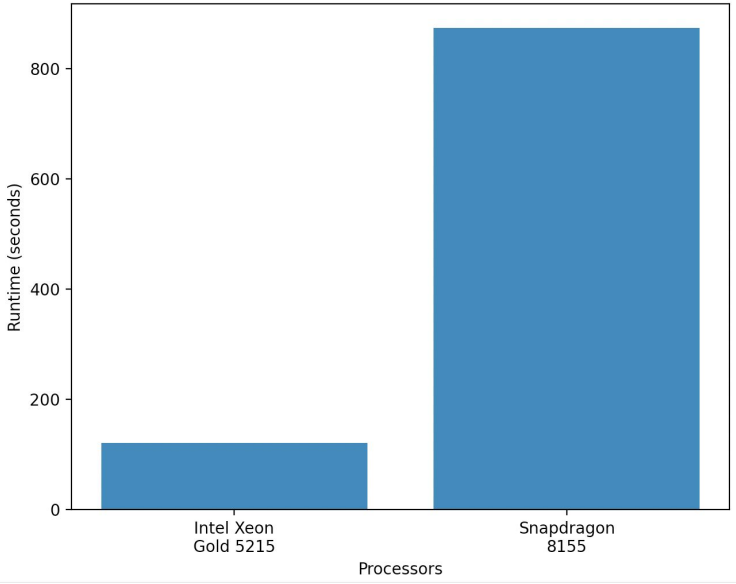
Image from https://earth.jpl.nasa.gov/emit/instrument/overview/

Cores	Runtime (s)	MSamples/sec	Current (mA)	Power (W)
1	2.06	9.56	600	2.4
2	2.33	16.91	850	3.3
4	2.42	32.56	1400	5.5
6	3.86	30.62	1500	6
8	3.62	43.54	1500	6

Match Filters

- Running on images from the Airborne Visible / Infrared Imaging Spectrometer (AVIRIS)
  - Images of Cuprite site in 2014
- Currently running only Kaolinite, Calcite and Alunite detection
  - Plan to expand to ideally ~20 minerals
- Runs on ARM only, single threaded
- Planning to expand to do mineral detection on Lunar images as well
- Runtime on the left are 8 images on 1 mineral
- Much of the runtime is I/O, so multithreading isn't as helpful immediately as we would like

https://www.jpl.nasa.gov/missions/airborne-visible-infrared-imaging-spectrometer-aviris



Mars Reconnaissance Orbiter (MRO) landmark classification

- Classifies landmarks from images taken from the High Resolution Imaging Experiment (HIRISE) instrument on MRO using a Convolutional Neural Network

• HIRISE Net:

	CPU/ARM	GPU/Adreno	DSP (Compute)	NPU
Total Power Utilization	5.7W	3.1W (~0.5x ARM)	2.1W (~0.37x ARM)	1.8W (~0.3x ARM)
		1.8X	32%	14%
Subsystem Utilization	100%	35%	85%	83% (CDSP)
Quantized Runtime (per image)	87.7ms	16.5ms (5x faster than ARM)	7.6ms (11x faster than ARM)	≈ 7.5ms (11x faster than ARM)
		5.3X	2.2X	
Energy Consumption (per image)	0.5 J	0.0512 J	0.0160 J	0.0135
		9.8X	3.2X	16%

- Ran many more machine learning models, this one is provided as an example
- Machine Learning Benchmarking discussed in more detail in another presentation
  - **Benchmarking Machine Learning on the Myriad X Processor Onboard the ISS**
    - Dr. Emily Dunkel

Future Work

- Measure additional CPU performance metrics
  - Power/Energy Consumption
  - RAM Footprint
- Benchmark on additional processors:
  - LEON4 Sabertooth, LEON3 Sphinx, RAD/PPC 750, and more.
- Parallelize Applications across multiple cores
- Make use of hardware acceleration
  - GPU, DSP, NPU
- Port more applications - see next slide

Future Planned Applications

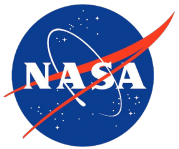
- Normalized Difference Index Science Product Generation
- Stereo Vision
- Decision Trees
- Campaign-Aware Path Generator (Pathogen)
- ECOSTRESS Data Processing Pipeline
- EMIT Science Data Processing Pipeline
- Sensorweb Tasking
- Lunar Flashlight NEAS (Near-Earth Asteroid Scout)
- Cloud Avoidance Algorithm
- etc

Your Application Here!

- Do you have applications that you would like to run on a Snapdragon 855 on the International Space Station? Contact Us! (Evaluation based on our interest in similar applications)
  - faiz.mirza@jpl.nasa.gov
- Requirements:
  - C/C++ code with a CMake build system
    - If any external libraries are used, we also need the source and a cmake file for that
    - If you want to use hardware acceleration, i.e. GPU, DSP, or NPU, contact us to discuss the feasibility.
  - OR
  - Python code (MAYBE)
    - Porting Python is hit or miss. If you have an interest in a python application, we can discuss it.

Conclusion

- Ported various planning, instrument processing, and machine learning applications
- Benchmarked problems on the Qualcomm Snapdragon 855
- Working towards measuring runtime performance against other flight processors
- Working towards measuring other metrics (power/energy, RAM footprint)
- Executed benchmarks on 855 onboard the ISS
- Work Intended to facilitate future flight of these capabilities to enable future single and networked autonomous spacecraft missions.



Jet Propulsion Laboratory
California Institute of Technology

jpl.nasa.gov