# Hyperspectral Image Compression & Decompression On FPGAs

**Team HEALTH (Hyperspectral Earth Acquisition of Longwave Thermal Heat)**

Brytni Richards
Dylan Vogel
Lorna (Xi) Lan

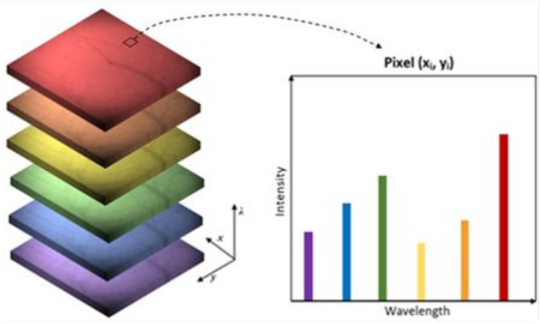
# Overview of Topic

- What is it
  - 3D hyperspectral image compression and decompression pipeline
  - FPGAs accelerate image compression/decompression
- Why is it important
  - Images may require many KB/MB of space
  - Project aiming for use in a small satellite payload
    - Real-time constraints
    - Size/memory constraints

## How are hyperspectral images different from normal images?

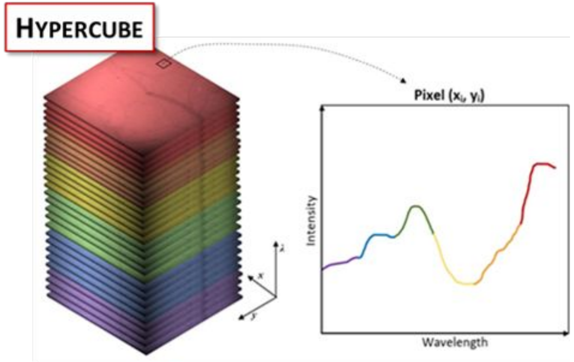
### Multispectral imaging

- N separated bands




### Hyperspectral imaging

- Continuous spectrum

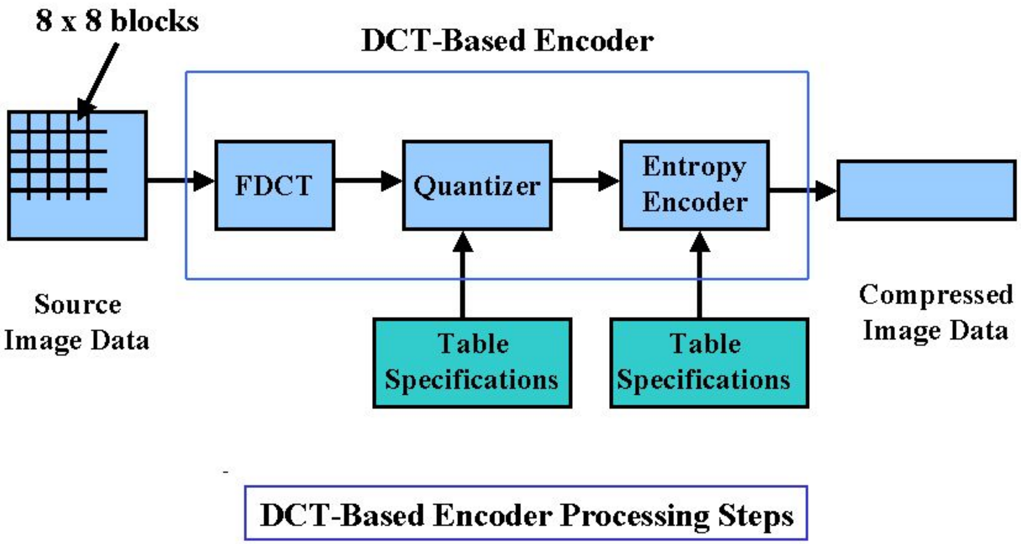



### Enables spectral analysis

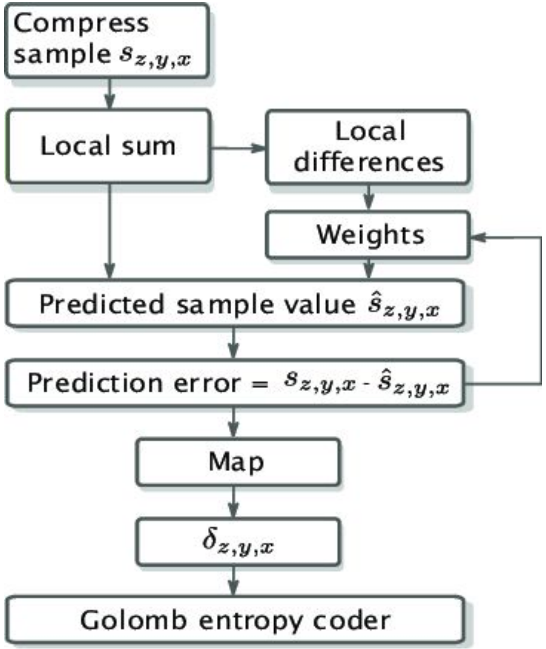
- Segmentation
- Spectral unmixing
- Evolution of spectra in time

# Compression Algorithms

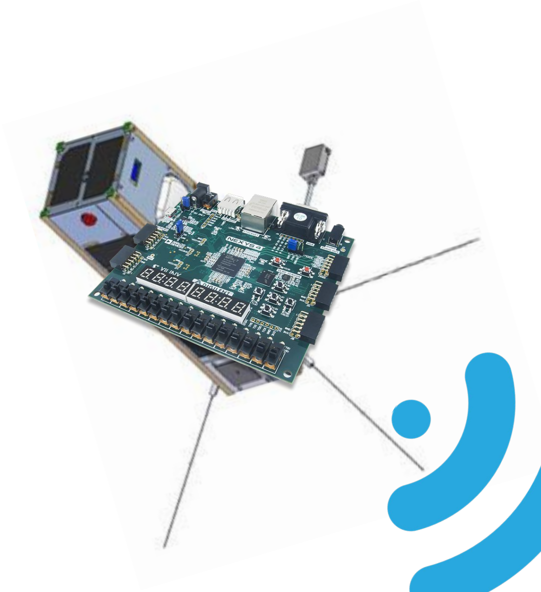
## DCT




## CCSDS-123

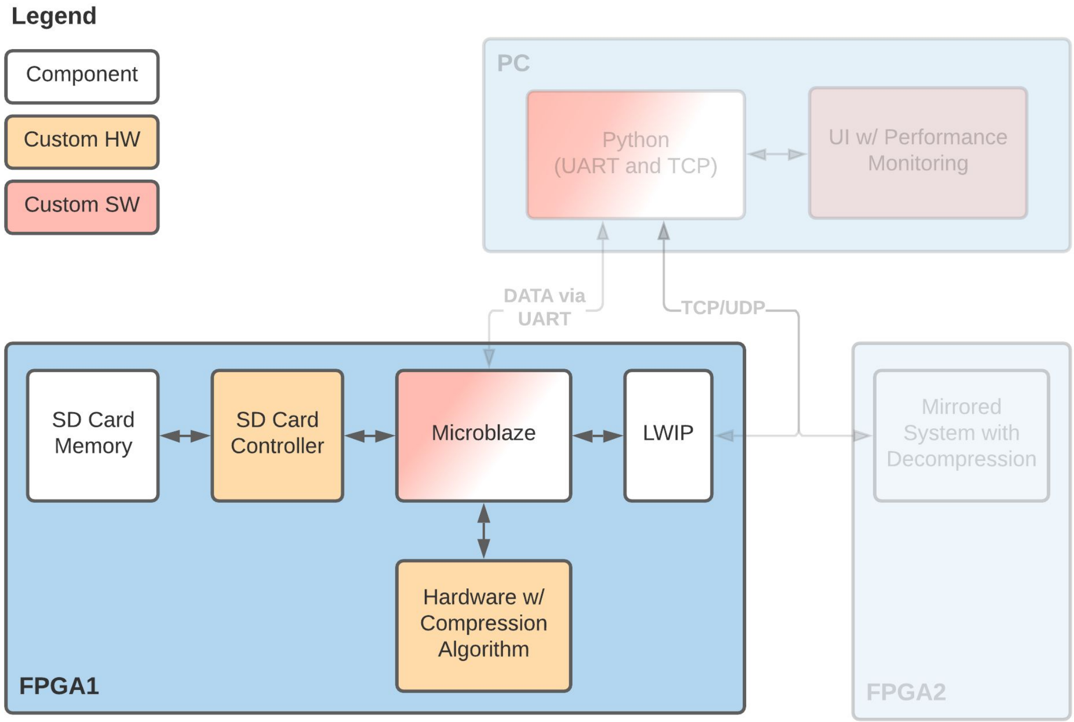


# Why Multi-FPGA?



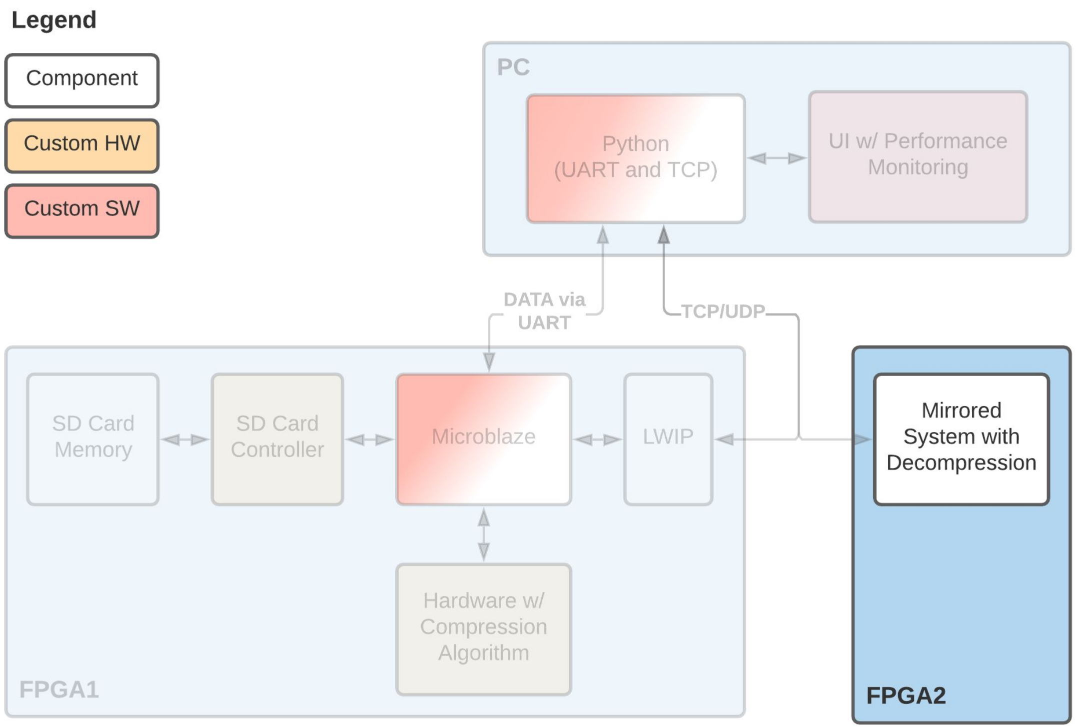
# Proposed System Block Diagram

- Core of the system is an FPGA compression/decompression block using DCT (baseline) or CCSDS-123.0-B-2 (exceed)
- SD card contains images from compression/decompression
- Microblaze processor runs the whole show and manages connectivity. Some custom and some existing code.



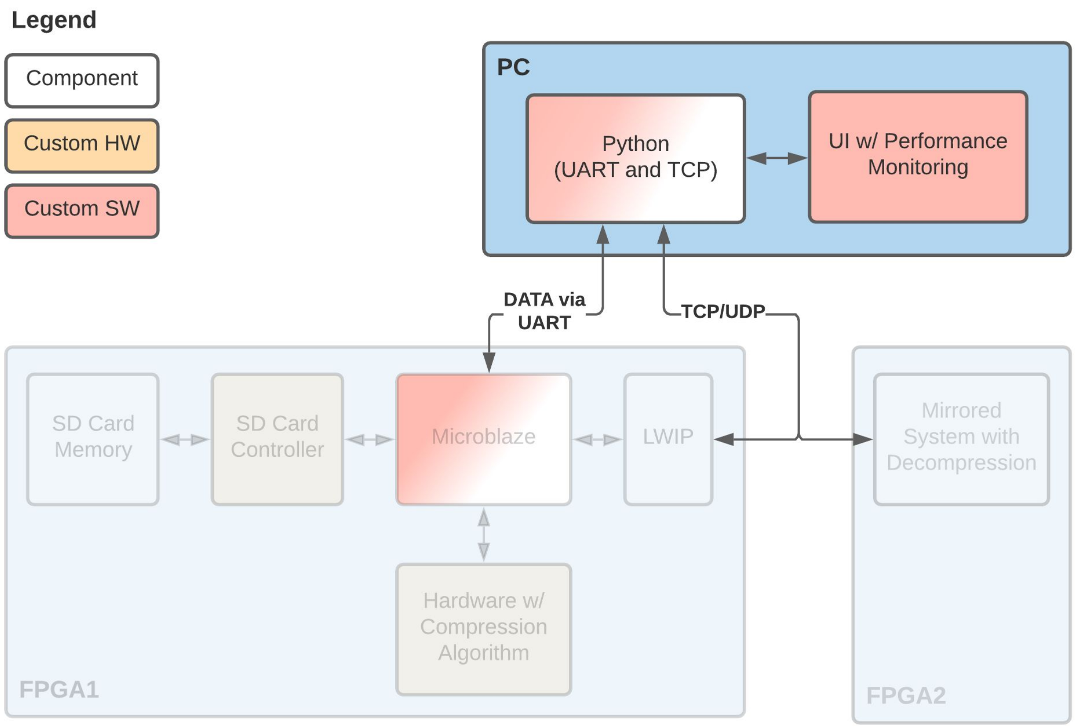
# Proposed System Block Diagram

- Decompression stage is a mirror of the first, although with the complementary algorithm
- Both FPGAs are collecting hardware performance monitoring data including throughput and  compression ratio.



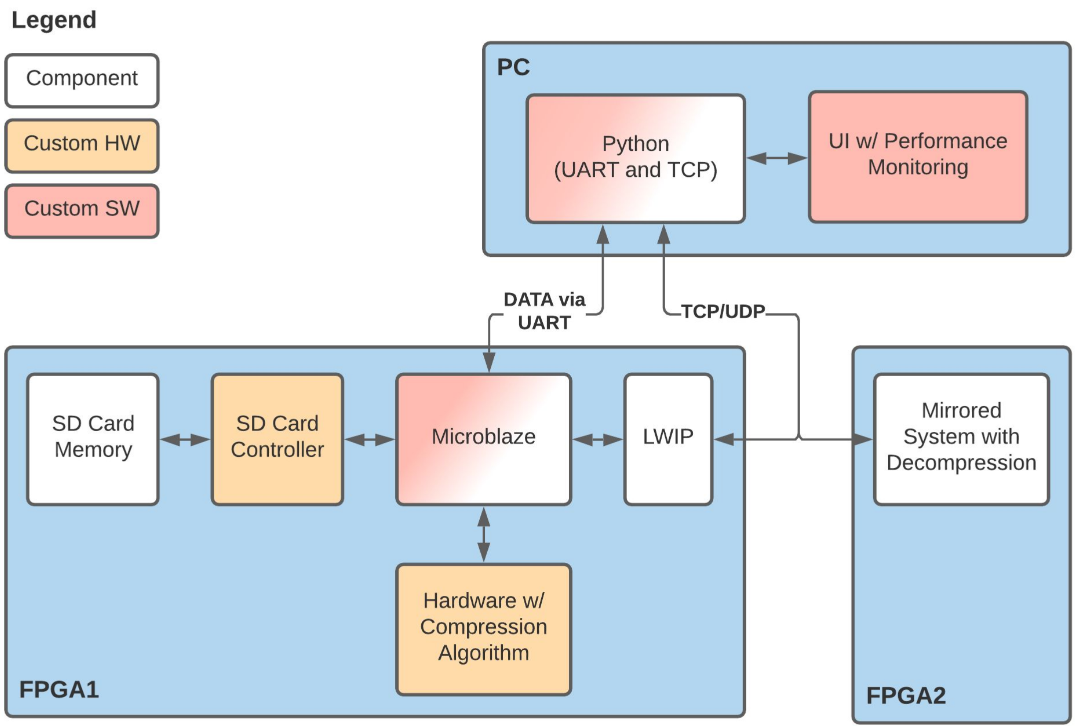
# Proposed System Block Diagram

- Host PC controls both FPGAs at a high level via TCP/UDP
- Host PC also sends/receives images from each FPGA
- Performance monitoring data is transferred from FPGAs and visualized in UI



# Proposed System Block Diagram

- Entire system simulates a remote sensing satellite mission with RF link bandwidth constraints and large data cubes
- FPGA-FPGA TCP/UDP communications  stands in for the RF link
- Host PC mimics a TT&C link which would also be present in a real system
- PC collects and visualizes performance data to help development



# Implementation Plan

**Brytni:** SD card implementation & data handling, 25% of performance visualization

**Dylan:** Image compression/decompression blocks, 75% of performance visualization

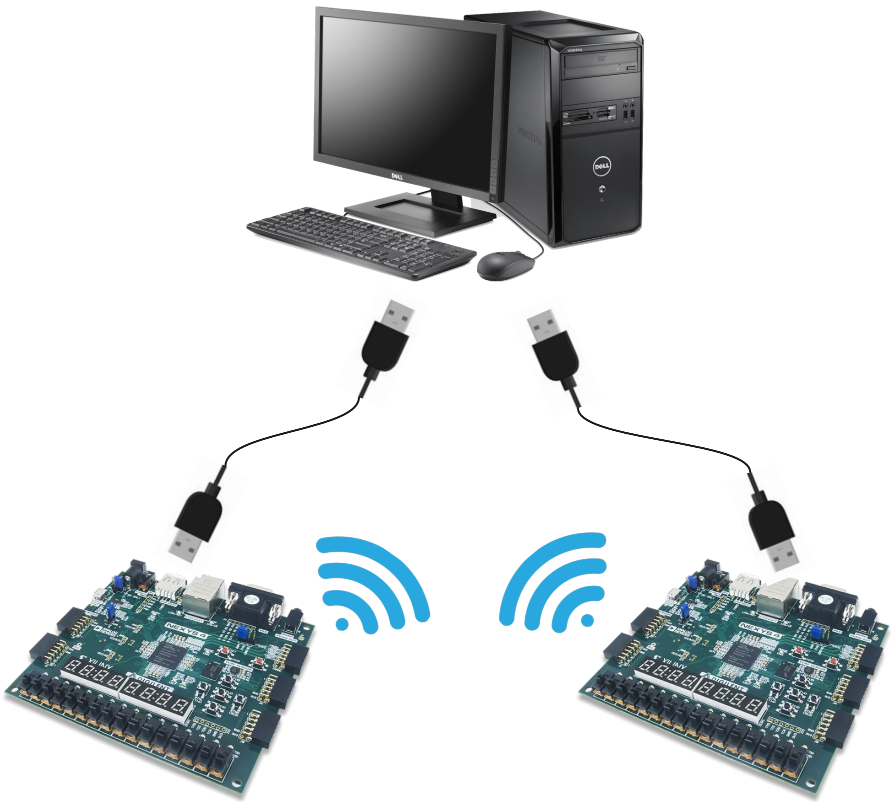
**Lorna**: Network connectivity (FPGA-FPGA and FPGA-PC TCP/UDP, UART to PC)

**All:** System integration (hard)

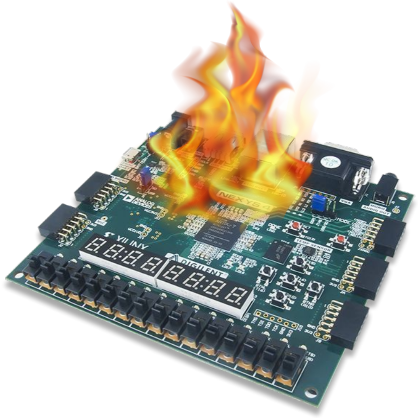
| Milestone # | High-Level Target |
|---|---|
| 1 | Research phase |
| 2 | SDIO/SPI physical layer implementation, DCT running in testbench, PC to FPGA via UART |
| 3 | SD card testbench, visualization started, initial network implementation done |
| 4 | Begin integrating SD card, microblaze, and compression/decompression blocks. |
| 5 | Finish SD card image transfer and network pipeline, pull performance monitoring stats |
| 6 | Road to final systems integration, time for exceed goals |

# Testing and Integration Plan

1. Compression/Decompression Algorithm
   a. Speed (bits per sample)
   b. Losses (absolute pixel intensity difference)
2. Networking
   a. Host PC to FPGA connectivity
   b. FPGA to FPGA connectivity
3. Performance Monitoring
   a. Basic terminal commands
   b. fAnCY GUI
4. SD Card
   a. Can write to it
   b. Can read from it



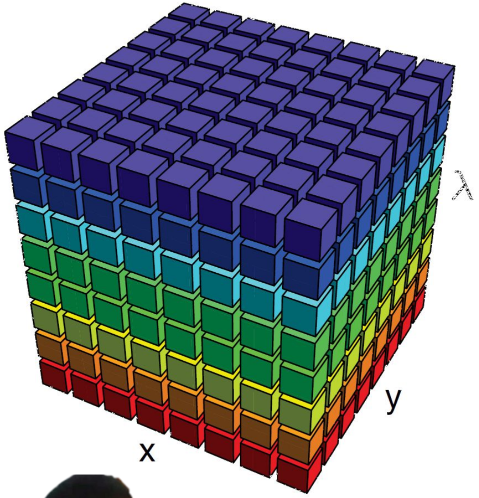
# Risks & Mitigation Plan

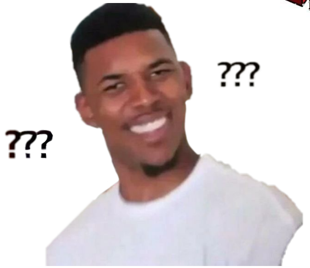


- Human Resources
  - In case of schedule conflict: Plan ahead and according to expected workload
  - In case of drop-out: Blood pact to pull through together
- Hardware Failures
  - In case of single FPGA failure: Test early, switch stations and inform TAs the abnormality
  - Insufficient RAM/LUTs: Test early, track usage space, de-scope image quality or algorithm complexity if necessary
- Software Failures
  - Infinite Loops: Be very careful with blocking operations. Consider interrupt-based event loop or real-time OS
  - Unknown Bugs: Reach out to professor and TAs early. Switch to another similar IP or algorithm.
- Network Failures
  - DESL network too slow: Test early and figure out the bandwidth of both wired/wireless networks. If still too slow just queue up image and demo one image transfer at a time
  - Lost access to DESL: Have a backup copy in the clouds or our personal machines

# Uncertainties

1. The software structure
    a. **Event loop** is simple to implement but beware of blocking operations
    b. **RTOS** has nice threading and locking mechanism, but more complex
2. Compression algorithms
    a. **DCT** is a simple starting point, working with only real numbers. However, it's quite bad in terms of compression loss
    b. **CCSDS** more complex, but is lossless and "industry standard" for space applications/remote sensing
3. Which dataset to test?
    a. Variety in image sizes, resolutions, bit-depths, band numbers and etc.
    b. Should the program be adaptive to various input? (auto-detect image size and quality etc.)
    c. Starting with this image set from Columbia University (CAVE Project)



# Q&A Portion

Any and all questions welcome.