Real-time Airborne Demonstration of Fast Lossless Hyperspectral Data Compression System for AVIRIS-NG and PRISM

Didier Keymeulen, Huy Luong, Nazeeh Aranki, Charles Sarture, Michael Eastwood, Ian Mccubbin, Alan Mazer, Matt Klimesh, Robert Green, David Dolman (3), Alireza Bakhshi (2)

Jet Propulsion Laboratory
California Institute of Technology
(2) B&A Engineering Inc.
(3) Alpha Data Inc.
Outline

• Overview of Fast Lossless (FL) Hyperspectral Data Compression Algorithm
• Fast Lossless FPGA Implementation
• Airborne Demonstrations
Fast Lossless (FL) MSI/HSI Compressor

FL Compressor Overview

![Diagram of FL Compressor](image)

**Approach:** Predictive compression, encoding samples one-at-a-time

- **Predictor**
  - Computes predicted sample value from previously encoded nearby samples (prediction neighborhood illustrated at right)
  - Adaptively adjusts prediction weights for each spectral band via adaptive linear prediction

- **Entropy Coder**
  - Losslessly encodes the *difference between predicted and actual sample values*
  - Adaptively adjusts to changing prediction accuracy

Direction of flight: 3
Cross-track 2
Band 1
3D neighborhood used for prediction.
Compression Algorithm: Estimation

- **Purpose**: Estimate a desired signal $d_t$ from an input vector $u_t$ using a linear estimator that is adaptively updated from previous results.

- **Compression of Estimate Error**:
  - Form estimate: $\hat{d}_t = w_t^T u_t$
  - Calculate estimation error: $e_t = \hat{d}_t - d_t$
  - $e_t$ is encoded in the compressed bitstream.
  - Update filter weights using the sign algorithm: $w_{t+1} = w_t - \mu u_t \text{sgn}(e_t)$ where $\mu$ is the “adaptation step size” parameter.

- **Naive approach**: use local neighborhood to construct $u_t$ around $d_t = s_0$

with $d_t = s_0$ and $u_t = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_5 \\ s_{10} \\ s_{15} \end{bmatrix}$

But performs poorly …. 

The samples are labelled $S_0, K, S_{19}$

3D neighborhood used for prediction.

Cross-track

Direction of flight

previous three bands

current band

current sample
Compression Algorithm: Local Mean Subtraction

- **Our solution:** compute simple preliminary estimates \( \hat{y}_t \) in each band at the spatial location of the sample being predicted, and subtract from the input samples.

\[
\begin{align*}
\hat{s}_{15} &= (s_{16} + s_{17} + s_{18} + s_{19})/4 \\
\hat{s}_{10} &= (s_{11} + s_{12} + s_{13} + s_{14})/4 \\
\hat{s}_5 &= (s_6 + s_7 + s_8 + s_9)/4 \\
\hat{s}_0 &= (s_1 + s_2 + s_3 + s_4)/4 \\
\end{align*}
\]

Use \( \hat{u}_t = \left[ \begin{array}{c} s_1 - \hat{s}_0 \\ s_2 - \hat{s}_0 \\ s_3 - \hat{s}_0 \\ s_5 - \hat{s}_5 \\ s_{10} - \hat{s}_{10} \\ s_{15} - \hat{s}_{15} \end{array} \right] \) and \( d_t = s_0 - \hat{y}_t \)

to compute the estimate \( \hat{d}_t = w^T u_t \) and the estimate error \( e_t = d_t - \hat{d}_t \).
Compression Algorithm: Implementation

- **Sign algorithm** is used for weight adaptation
- **Estimation error** is encoded using Golomb power-of-2 codes
- **Dataset is divided into parts (32 lines each)**, which are compressed independently. This provides some error containment.
- **Each spectral band has its own prediction weights**, maintained independently of the prediction weights for other spectral bands.
Compression Algorithm: Other Methods

Compare our “Fast lossless” compression algorithm with:

- **ICER-3D**: a 3-D-wavelet-based compressor which is the state-of-the-art (ICER-2D is used on both spirit and opportunity MER rovers)

- **Rice/USES (GSFC)**: algorithm used in USES chip, with the multispectral predictor option.

- **JPEG-LS**: is most efficient for 2D and is applied to the spectral bands independently

Other Methods:

- **Differential JPEG-LS**: JPEG-LS applied to the differences between the successive spectral bands

- **SLSQ and SLSQ-OPT**: two versions of Spectral-oriented Least Squares (SLSQ) [Rizzo et al., 2005]. Algorithms with complexity roughly similar to that of ours.

- **3-D CALIC**: a nontrivial extension of the basic (2-D) CALIC algorithm to multispectral imagery. More complex.

- **M-CALIC**: multiband CALIC, another extension of CALIC to multispectral imagery. More complex.

- **ASAP**: Adaptive Selection of Adaptive Predictors [Aiazzi et al., 2001]; more computationally intensive than any of the other compressors in the tables
Comparison using Aviris Data Sets Test Bed

Moffett Field (vegetation, urban, water)
Cuprite (geological features)
Jasper Ridge (vegetation)
Lunar Lake (calibration)
Low Altitude (high spatial resolution)

AVIRIS data sets represent different scenes
Tests using 19 uncalibrated AVIRIS data sets:
• original sample size: 12 bits/sample
• data size: \((614 \times 512)\) pixels \(\times\) 224 bands

Methods:
JPEG-LS: is most efficient for 2D; GSFC/USES use chip; ICER-3D SOA (ICER-2D MER rovers)
Compression Algorithm Features

• **Performance:** outstanding compression effectiveness

• **Robust:** requires no training data or other specific information about the nature of the spectral bands for a fixed instrument dynamic range

• **Simple:** well-suited for implementation on FPGA hardware and easily parallelizable

• **Low computational complexity.** required operations per sample are:
  – 6 integer multiplications
  – 25 integer addition, subtraction, or bit shift operations
  – Golomb coding operations

• **Modest memory requirement:** enough to hold one spatial-spectral slice of the data (e.g., ≤650 Kbytes for AVIRISng data with 481 bands and 640 samples/line)

• **Instrument:** well-suited to push broom instruments
JPL Lossless Data Compression is a CCSDS Standard

The Consultative Committee for Space Data Systems (CCSDS) Multispectral & Hyperspectral Data Compression working group has adopted the FL compressor as international standard CCSDS-123.0-B-1

FL verification software has demonstrated outstanding performance on all of the myriad airborne and spaceborne imagers represented in the CCSDS test data set:

- Hyperspectral imagers:
  - AVIRIS, Hyperion, SFSI, CASI, M3, CRISM
- Ultraspectral sounders:
  - AIRS, IASI
- Multispectral imagers:
  - MODIS, MSG, PLEIADES, VEGETATION, SPOT5
High Speed FL Implementations: CPU/GPU

- FL is well-suited for high-speed parallel implementations:
  - **GPU: 7× speed-up** – A GPU hardware implementation targeting the current state-of-the-art GPUs from NVIDIA®: mobile version GTX560M and desktop version GTX580
  - **OpenMP: 3× speed-up** – A 12-core implementation targeting the mobile Intel® quad-core i7™ processor and the desktop Intel® hexa-core Xeon™ processor
- Example: uncalibrated AVIRIS hyperspectral image (137MBytes)
  - Compression time: 11.38 sec on single-core CPU, 3.68 sec on 12-core CPU, and 1.57 sec on GPU

<table>
<thead>
<tr>
<th>Implementation</th>
<th>Speedup</th>
<th>Time (s)</th>
<th>Speed (Mbit/s)</th>
<th>Speed (MSamp/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU GeForce GTX 580</td>
<td>725%</td>
<td>1.57</td>
<td>583.08</td>
<td>44.85</td>
</tr>
<tr>
<td>GPU GeForce GTX 560M</td>
<td>596%</td>
<td>1.91</td>
<td>479.29</td>
<td>36.87</td>
</tr>
<tr>
<td>GPU Tesla C2070</td>
<td>486%</td>
<td>2.34</td>
<td>391.21</td>
<td>30.09</td>
</tr>
<tr>
<td>Dual Hex Core (12 cores)</td>
<td>309%</td>
<td>3.68</td>
<td>248.76</td>
<td>19.14</td>
</tr>
<tr>
<td>Dual Hex Core (8 cores)</td>
<td>272%</td>
<td>4.19</td>
<td>218.48</td>
<td>16.81</td>
</tr>
<tr>
<td>Dual Hex Core (4 cores)</td>
<td>259%</td>
<td>4.39</td>
<td>208.53</td>
<td>16.04</td>
</tr>
<tr>
<td>Quad Core (4 cores)</td>
<td>196%</td>
<td>6.87</td>
<td>133.25</td>
<td>10.25</td>
</tr>
<tr>
<td>Dual Hex Core (1 core)</td>
<td>115%</td>
<td>9.9</td>
<td>92.47</td>
<td>7.11</td>
</tr>
<tr>
<td>Quad Core (1 core)</td>
<td>100%</td>
<td>11.38</td>
<td>80.44</td>
<td>6.19</td>
</tr>
</tbody>
</table>

Data Rate:
AVRISng (481*640 pixels per frames @100 frames/sec): 500Mbit/s
Future (481*1600 pixels per frames @100 frames/sec): 1300 Mbit/s
FPGA FL: 640 Mbit/s
High Speed FL Implementations: CPU/GPU

Version 2: Even faster with re-designed data path

- Redesigned data path implementation: Parallel computation across multiple 32 frames of the full image
- Total speed-up for Version 2
  - **GPU: 56× speed-up** – 137MB AVIRIS image compression time: 204 ms (vs. 11.38 sec)
  - **12-core CPU: 20× speed-up** – 137MB AVIRIS image compression time: 569 ms (vs. 11.38 sec)
- True real-time performance (2×-5× real-time target of 800Mb/s or 50MSamples/sec) BUT require 100 Watt

<table>
<thead>
<tr>
<th>Version</th>
<th>Time (ms)</th>
<th>Throughput (Mb/s)</th>
<th>Throughput (MSamp/s)</th>
<th>Speedup vs. V1</th>
</tr>
</thead>
<tbody>
<tr>
<td>OMP - v1 - 8 core</td>
<td>4488</td>
<td>194.53</td>
<td>14.96</td>
<td>1.00</td>
</tr>
<tr>
<td>OMP - v4 – 12 core</td>
<td>569</td>
<td>1534.68</td>
<td>118.05</td>
<td>7.89</td>
</tr>
<tr>
<td>CUDA - v1</td>
<td>1910</td>
<td>457.08</td>
<td>35.16</td>
<td>1.00</td>
</tr>
<tr>
<td>CUDA - v4 - 1 GPU</td>
<td>226</td>
<td>3862.97</td>
<td>297.15</td>
<td>8.45</td>
</tr>
<tr>
<td>CUDA - v4 - 2 GPU</td>
<td>204</td>
<td>4279.56</td>
<td>329.20</td>
<td>9.36</td>
</tr>
<tr>
<td>Decompress (serial)</td>
<td>3585</td>
<td>243.53</td>
<td>18.73</td>
<td>1.00</td>
</tr>
<tr>
<td>Decompress (parallel)</td>
<td>857</td>
<td>1018.16</td>
<td>78.32</td>
<td>4.18</td>
</tr>
</tbody>
</table>

Data Rate:
- AVRISng (481*640 pixels per frames @100 frames/sec): 500Mbit/s
- Future (481*1600 pixels per frames @100 frames/sec): 1300 Mbit/s
- FPGA FL: 640 Mbit/s
**FL FPGA: ARTEMIS & AVIRIS-NG**

**FL FPGA Compression IPs** for whiskbroom and pushbroom imagers

- **Xilinx Virtex-4 Lab Demonstration for ARTEMIS**
  - Implemented on Xilinx Virtex4 ML401 prototype board.
  - 17 MB image data (32 frames) uploaded serially to 256 DDR SDRAM prior to compression

- **Xilinx Virtex-5 Real-Time Airborne Onboard Compression**
  - Implemented pushbroom compressor on COTS Virtex 5 (equivalent to V5 Rad-hard device). Compresses one sample every clock cycle, a speed of 40 MSample/sec
  - Implementation tested in National Instruments PXI environment which includes a PXIe-7962R board with Xilinx Virtex-5 SX50T and two 256MBytes DRAMs. The system is connected to the airborne AVIRIS-NG HSI instrument and provides real-time onboard compression
Real-time aircraft onboard compression

- Implemented pushbroom FL compressor on a COTS Virtex 6. Compresses one sample every clock cycle, a speed of 40 MSample/sec.
- Implementation tested via Alpha-Data ADPE-XRC-6T which includes
  - Xilinx Virtex-6 LX240T
  - two 256MBytes DRAMs (32bits data word, 3.2GBytes/sec per bank)
  - PCIe x4 Gen2 (500MBytes/sec per lane).
- PRISM and AVIRISng HSI image data transferred in real-time (60MBytes/sec) to the Virtex-6 via Alpha-Data FMC-CLINK-MINI camera link board, compressed on the Virtex-6 and transferred through PCIe to a 1GBytes SSD drive configured as RAID0 (500MBytes/sec)
FL FPGA IP Main Block Diagram

- **CURRENT Z DOUBLE BUFFER (Zx1x2)**
- **THREE UPPER Y AND PREVIOUS Z PIXEL BUFFER(S) (Zx1x4)**
- **LOCAL MEAN**
- **WEIGHT**
- **MULTIPLIER**
- **DELTA**
- **ESTIMATE**
- **ENCODER**
- **PACKER**

**FLOW:**
- **UNCOMPRRESSED DATA IN** to FPGA
- **EXTERNAL RAM**
- **LOSSLESS COMPRESSION ALGORITHM**
- **LOSSLESS COMPRESSED DATA OUT**
FL FPGA Architecture

Alpha-Data ADPE-XRC-6T

- Camera Link interface
- Resync Pass through
- BIL to BIP Formatting
- Camera Link & OCP interface
- FL FPGA Architecture
- Xeon CPU
- PCIe Gen2 X4 0.5GB/s
- RAM
- SSD 1 TB 0.5GB/s (raw Compressed)
- Compression
- DMA Bank#2 transfer
- DMA Bank#1 transfer
- OCP Interface Mux SDRAM #1
- OCP Interface Mux SDRAM #2
- DDR bank #1 SDRAM 512MB 32bits; 3.2 GB/sec
- DDR bank#2 SDRAM 512MB 32bits; 3.2 GB/sec

Virtex6-LX240T-3

- Custom App (JPL)
- Control & Status
- Hyper-spectral source
- FL Compression
- Hyper-spectral sink

Alpha-data FMC-CLINK CameraLink
- 640x285, 165Hz, 60MB/s BIL; 16 bits/sample

IMU/GPS

Host
- Software
- Drivers
- Xeon CPU
- RAM
- SSD 1 TB 0.5GB/s (raw Compressed)
## FL FPGA Resource Utilization – Virtex6

### Device Utilization Virtex6-LX240T-3 (Compressor and Interface)

<table>
<thead>
<tr>
<th></th>
<th>Available</th>
<th>Used</th>
<th>Utilization All</th>
<th>Utilization Compressor</th>
<th>Utilization Virtex5 Compressor (estimate)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slice Register (Flip-Flop)</td>
<td>301,440</td>
<td>37,284</td>
<td>12%</td>
<td>4%</td>
<td>8%</td>
</tr>
<tr>
<td>Slice Look-up-table (LUTs)</td>
<td>150,720</td>
<td>37,374</td>
<td>24%</td>
<td>8%</td>
<td>8%</td>
</tr>
<tr>
<td>Fully used LUT-Flip Flop pairs</td>
<td>50,693</td>
<td>19,105</td>
<td>38%</td>
<td>13%</td>
<td>26%</td>
</tr>
<tr>
<td>Block RAM/FIFO</td>
<td>416</td>
<td>108</td>
<td>25%</td>
<td>12%</td>
<td>12%</td>
</tr>
<tr>
<td>DSP 48eS</td>
<td>768</td>
<td>6</td>
<td>1%</td>
<td>1%</td>
<td>1%</td>
</tr>
</tbody>
</table>

### Device Utilization SDRAM (AVIRISng)

<table>
<thead>
<tr>
<th></th>
<th>Available</th>
<th>Used</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDRAM Bank#1 (2 segments)</td>
<td>256 MBytes</td>
<td>40 MBytes</td>
<td>20%</td>
</tr>
<tr>
<td>SDRAM Bank#2 (3 segments)</td>
<td>256 MBytes</td>
<td>60 MBytes</td>
<td>24%</td>
</tr>
</tbody>
</table>

### Timing: Critical Path

<table>
<thead>
<tr>
<th>Block</th>
<th>Critical Path Timing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchronization frames with IMU/GPS</td>
<td>&lt;25ns</td>
</tr>
<tr>
<td>Transpose BIP to BIL</td>
<td>&lt;10ns</td>
</tr>
<tr>
<td>Predictor</td>
<td>12.070 ns</td>
</tr>
<tr>
<td>Entropy Encoder</td>
<td>10.029 ns</td>
</tr>
<tr>
<td>Packer</td>
<td>7.377 ns</td>
</tr>
</tbody>
</table>

The implementation compresses one sample every clock cycle, which results in a speed of 40 MSample/sec.
Comparison during airbone AVIRISng mission (June 2014)
Comparison during airborne AVIRISng mission (June 2014)

DataSets

- Kingsburg, Agriculture Field (13,000 ft)
- San Joaquin (8,000 ft)
- Sierra (17,500 ft)
- Soda Lake (5,000 ft)
- Fresno, Agriculture Field (10,000 ft)

Compression Ratio

- Fast Lossless 12bits/sample
- Fast Lossless 13bits/sample

Original sample size: 14 bits/sample
Data size: 640 cross track by 481 bands
Summary

We presented an FPGA implementation of a novel hyperspectral data compression algorithm and its flight demonstration: JPL adaptive Fast Lossless compressor.

The implementation targets the Xilinx Virtex FPGAs and provides an acceleration of at least 7 times the software implementation on a single core of the Intel® Hex Core™ i7, making the use of this compressor practical for satellites and planet orbiting missions with hyperspectral instruments.

Future development will provide multiple implementations and near lossless data compression for accommodating large Focal Plane Array (FPA). We will also develop options to deploy various versions of the algorithm to accommodate data from different instrument types as well as radiance and reflectance data. And finally explore new hardware technologies such as System-on-the-Chip (SoC) to embed the compression next to the FPA ROI and fast I/O interface to the instrument (e.g. optical).