WHITE PAPER

How modern multi-core space processors increase data throughput with lossless compression.

June 2024

TELEDYNE | Teledyne e2v Semiconductors

ABSTRACT

Teledyne e2v reviews in-house benchmarking that validates the ability of modern processors to implement spaceborne lossless compression. This shows that system level economic and performance benefits emerge from using multi-core communication processors in combination with modern compression algorithms capable of handling a variety of file formats.

Space missions, where bandwidth and storage capacity are always economically constrained, demand efficient data management and transmission techniques. Ideally, system enhancements may yield from applying effective data compression that conserve essential information while minimizing data set size. Compression demands that data be stored locally and processed (i.e. encoded) onboard before being transmitted to Earth. Advanced lossless algorithms place strong demands on spaceborne computational power. Thankfully, data compression algorithms together with processor performance have advanced considerably over the last few decades. As you will see, it is now possible for the latest, leading-edge, multi-core communication processors to deliver rapid real-time compression. Space grade processors, such as new devices from Teledyne e2v, prove highly effective at supporting this application.

The focus of this study is two ARM multi-core A72-based processors: the LS1046 and LX2160 communication processors, which are compared with equivalent industrial grade (i.e. non-Space) processors.

Article insights...

- Space missions benefit from efficient data compression and transmission due to limited bandwidth and restricted local storage
- Compression can now be run on multiple cores of the latest space-grade processors in real-time, enhancing communication link throughput
- Lossless compression methods such as Run-Length Encoding, Huffman and Lempel-Ziv-Welch (LZW) preserve data set fidelity
- Benchmark results provided compare a series of processors with compression rates achieved using the LZ4 lossless algorithm

Some of the practical aspects of implementing compression algorithms will be detailed along with an example of a popular lossless method (LZ4). Finally, you will benefit from the benchmarking comparison providing compression rate measured in MB/s to validate the capabilities of a range of standard, modern processing platforms.

INTRODUCING DATA COMPRESSION – WHAT IS IT?

The goal of data compression is to reduce dataset size, lower demands on local storage memory and communication bandwidth, helping to deliver economic and resilient missions. Compression removes redundant data to reduce hard system speed and performance constraints. Each space application has unique requirements. This work sought to understand these further, with a view to covering a broad spectrum of compression use cases.

There are two data compression approaches, both of which reduce the data set size for ongoing storage and transmission.

Lossy compression sees some data being discarded to achieve a certain compression ratio for optimal transmission or storage. Whilst lossy techniques can indeed provide predictable compression and thus a smaller file sizes, by nature it also leads to a reduction in data quality. Moreover, once compressed, the original data fidelity is lost. Examples of commercial lossy algorithms include JPEG (Joint Photographic Experts Group) and MPEG (Moving Picture Experts Group) formats. [3] Owing to its lossy nature, it is unsurprising that pro-photographers insist on maintaining their stock images as Raw (i.e. uncompressed & unprocessed) files.

Lossy Algorithm example (JPEG)

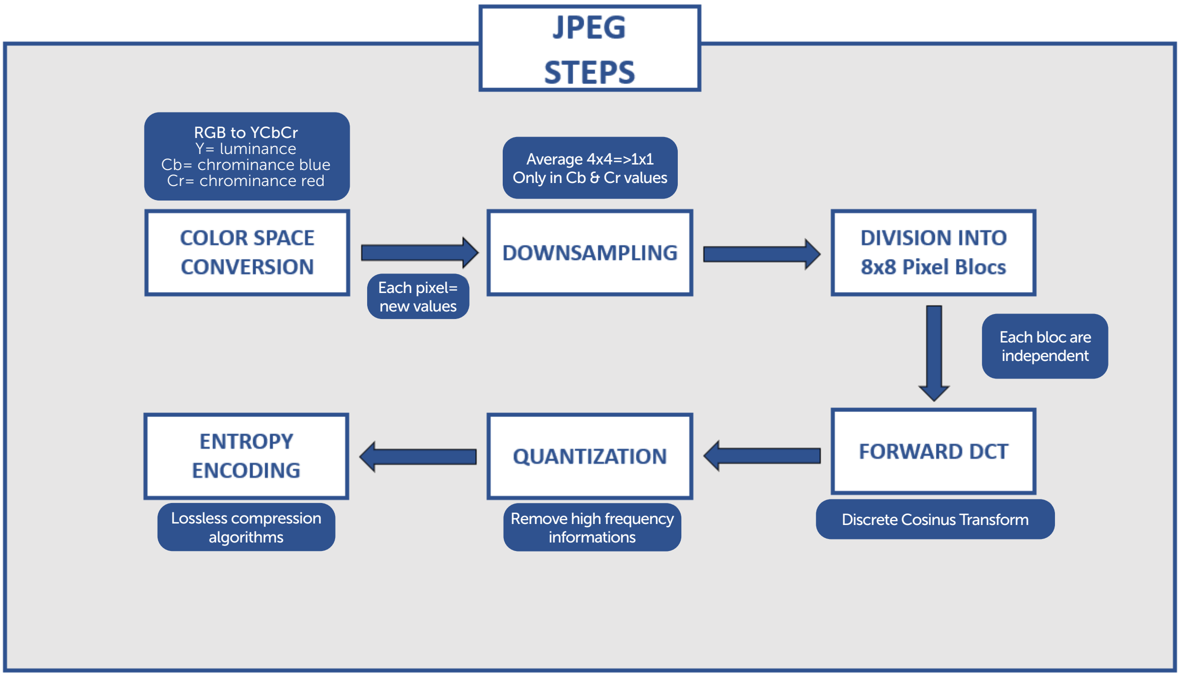

Figure 1 : JPEG compression (Lossy) [5]

An important consideration for satellite compression is power consumption. Compared to FPGAs, the communication processors highlighted, given their fine-line process geometries, have a noticeable power advantage in terms of MB/watt consumed. Thus, when lossy compression is to be used, processors are most likely the best solution.

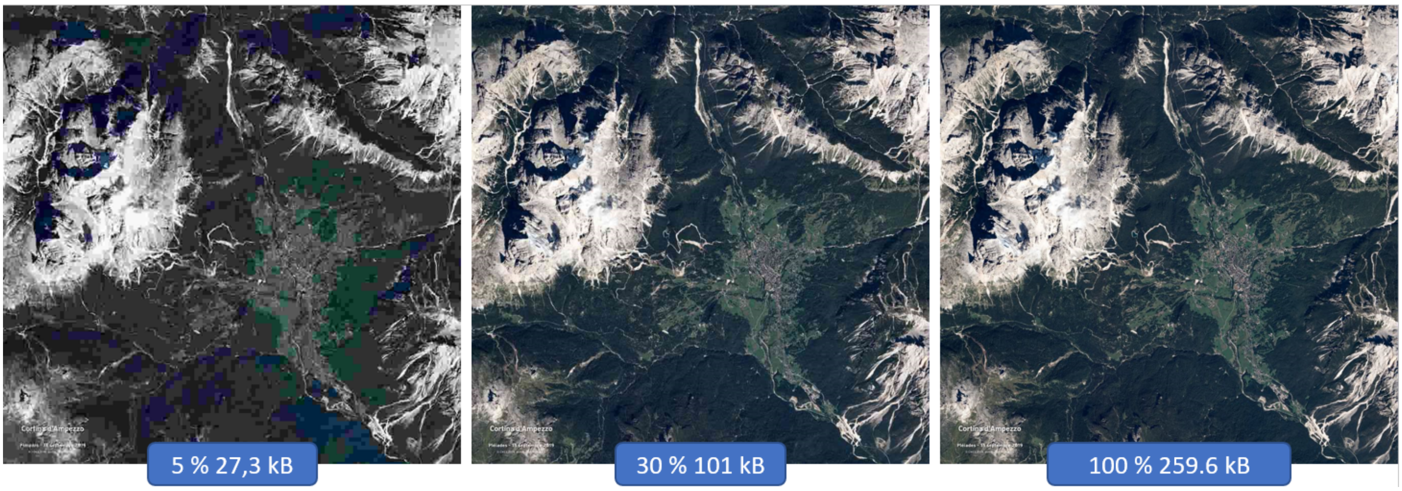

Figure 2 : Example of lossy compression effect [CNES pictures]

The image set (figure 2) shows examples of lossy JPEG compression applied at different compression ratios to the same image. Image quality (sharpness and resolution) increase with file size from left to right.
The alternative to lossy compression is lossless compression.

How modern multi-core space processors increase data throughput with lossless compression.

June 2024

TELEDYNE | Teledyne e2v Semiconductors

Lossless Compression are a family of compression methods preferred by professionals in a variety of fields, since files sizes can be reduced without loss of information. In fact, lossless compression is reversible and works by identifying and eliminating statistical redundancies within the data set.

Over the years a wide variety of lossless approaches have been advanced. The topic is complex and beyond the scope of this article, but these methods fall broadly into four categories of encoding types namely:

- Transform coding (e.g. Discrete cosine transform)
- Prediction-based coding (e.g. Adaptive PCM)
- Entropy coding (e.g. Huffman)
- Dictionary-based coding (e.g. LZW & LZ4)

Compression info box...
In information theory, data compression, source coding, or bit-rate reduction is the process of encoding information using fewer bits than the original representation.

A compression algorithm is either lossy or lossless. Lossless compression, preferred by professionals, reduces datasets by identifying and eliminating statistical redundancy.

Lossy compression reduces dataset size by removing unnecessary or less important information in the dataset. It renders an acceptable optimal quality at the recipient.

Common lossless algorithms include Run-Length Encoding (RLE), Huffman and Lempel-Ziv-Welch (LZW) encoding. These techniques are well-suited to professional spaceborne scenarios. [2]

The power of lossless compression lies in the ability to recover the original, pre-encoded file and it is anticipated that most compression use cases exploited in space, exploit lossless encoded data.

The following table provides a contextual summary of the two compression approaches highlighted so far.

Compression type	Lossy	Lossless
Example algorithms	JPEG, MPEG, MP3 etc.	RLE, Huffman, LZW, LZ4, ZIP
Primary objective of compression	Cost effectiveness	Data integrity
Data quality determined by	User/market	Application
Advantages	Cost effective trade-off supporting many consumer needs	Wide variety of methods suiting specific professional applications
Memory impact	Optimized for purpose	Variable - depends on algorithm
Transmission impact	Significant increase	Increased -depends on algorithm
CPU resources	Deployed at source, modest	Deployed both ends, high
Disadvantages	Permanent loss of data fidelity, potential creation of unwanted artifacts	Processing complexity, need for paired encode and decode capabilities

Table 1: Contextual summary of lossy and lossless compression methods.

BENCHMARK COMPRESSION ALGORITHM:

The benchmarking carried out by Teledyne e2v used the LZ4 compression plugin. LZ4 is a widely used lossless algorithm. It delivers a good compression ratio and offers fast decompression speeds, making it particularly suited to scenarios where fast data processing is essential. LZ4 stands for Lempel-Ziv 4, indicating its relation to the Lempel-Ziv family of algorithms. These are widely used across internet infrastructure and cloud networks to boost data throughput. They work well in terms of speed and simplicity but carry a penalty in respect of encoding memory and do not achieve the highest levels of compression on data sets, especially those with limited repetition.

Algorithmic example (LZ4)

Outline of LZ4 algorithm operation with respect to compression:

- The algorithm uses a sliding window to scan for repeated bit sequences (matches) and subsequent unique data patterns (literals) which are represented in the output data as tokens
- A hash table is used to identify matches with offsets and lengths, literals are stored 'as-is'
- The algorithm produces a compressed output comprising tokens each representing a literal sequence (unmatched data) or a match (referring to a previous sequence sent)

Note: The LZ4 compression symbol library is not empty at initiation, it is pre-configured with classic information, such as the alphabetic character set or alternatively the most recent block of data sent.

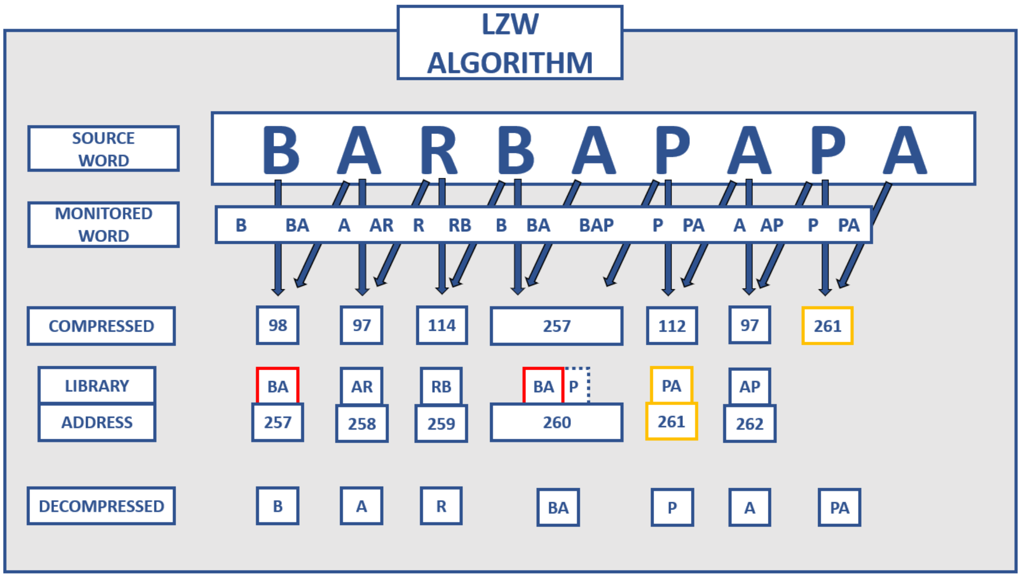

Figure 3: Contextual summary of lossy and lossless compression methods.

Figure 3 shows the encoding flow of the Lempel-Ziv-Welch method. It is designed to save decompression time. Thanks to the literals library, decompression could be six times faster than the compression phase. In some cases, there may be a benefit in creating a custom library optimized to specific applications.

LZ4 algorithm is often used because of its low latency. This includes real-time data processing and communication systems. It has become popular due to its compatibility with many programming languages and platforms.

Some plugins offer different compression rations influencing performance and compression speed. The higher the level, the greater the compression and therefore the lower the speed. For this study, the ninth level of LZ4 compression was selected as optimal for each of the benchmark file types.

The encoding (compression) process of LZ4 requires more processing resources compared to the decoding. This is due to the additional computational complexity involved in finding matches, managing the hash table, and making decisions about how to best compress the data. Decoding is simpler and thus faster, primarily involving the straightforward task of reconstructing the original data from the encoded tokens.

Figure 4: Example of lossless compression with LZ4 compression executed. [CNES picture]

INTRODUCTION SPACE-GRADE COMMUNICATION PROCESSORS

Consider a pair of powerful new, space grade communication processors.
The LS1046 and LX2160 are multi-core, 64-bit ARM Cortex-A72 processors. They are radiation tolerant designs manufactured by Teledyne e2v. They serve many applications requiring compute intensive capabilities. Notable examples include embedding artificial intelligence since they can run deep learning, AI algorithms. This aids image acquisition and processing. This approach delivers the real technical benefit of pre-processing source data at the ‘edge’, helping reduce downlink bandwidth. Other end applications include:

- Communication satellites, embedding AI and security
- Landing and avionics space systems, robotics and mechanical arm controls
- Human mission exploration, science missions
- Early warning, observation satellites - automated situation detection & awareness
- Meteorological satellites

What is a DMIP?

DMIP stands for «Dhrystone Million Instructions Per Second». It measures processor performance. It is derived from the 1980s Dhrystone benchmark, which tests general-purpose integer operations.

DMIP quantifies how many million instructions a processor can handle relative to a reference machine (VAX 11/780).

While useful for comparing computational performance, DMIP focuses on integer operations. Thus, it does not fully represent modern floating-point computational workloads.

This study aims to demonstrate the efficiency of Teledyne e2v communication processors in performing data compression for space applications.

Figure 5: LS1046

LS1046-Space is a Quad-core processor, running at speeds up to 1.8GHz delivering 30k DMIPS, ECC-protected L1 and L2 cache, and a rich set of peripherals such as packet processing acceleration, 1/10 Gb Ethernet, PCIe® Gen3, SPI, I^{2}C, UART.

Status: Fully qualified and released.
Qualified for Space: up to Level 1 (NASA EEE-INST-002 - Section M4 – PEMs & ECSS-Q-ST-60-13C).

Figure 6: LX2160

LX2160-Space is the latest 16-core processor, with up to 200k DMIPs, running at up to 2.2GHz. Featuring WRIOP that enables it to handle high-speed peripherals including 100 GbE, multiple PCIe Gen3.0, hardware L2 switching, DPAA2 with 100 Gbps decompression/compression, and an optimized 50 Gbps Encryption engine. This is also complemented by an array of popular standard data interfaces.

Status: LX2160 is currently under development.

PROCESSOR DESIGN FACTORS INFLUENCING COMPRESSION

There are several factors that influence processor compression performance beyond the specifics of the algorithm used. Knowledge of these factors will help guard against surprises when a specific processor appears to underperform, based upon raw DMIPs considerations. Perhaps the top two are hardware parallelism (aided by multi-core processor design) and then careful cache and local memory architecture. How and why do these factors dominate? There are clues in considering what compression algorithms need to achieve.

Parallelism and Multicore Architecture

Parallelism in both hardware and software design plays a crucial role. This includes:

- Multicore Processors: Compression algorithms often involve tasks that can be performed in parallel, such as processing different blocks of data simultaneously. Multicore processors significantly speed these tasks by distributing the workload across the cores.
- SIMD (Single Instruction, Multiple Data) Extensions: SIMD extensions like AVX (Advanced Vector Extensions) and SSE (Streaming SIMD Extensions) allow a single instruction to process multiple data points concurrently.

By leveraging multicore processors and SIMD capabilities, compression algorithms can achieve higher throughput and reduced execution time.

Cache Architecture and Memory Bandwidth

The efficiency of data access and movement within the system has a profound impact on compression speed. Key aspects include:

- Cache Size and Levels: Large and multi-level caches (L1, L2, L3) can store frequently accessed data and instructions physically closer to the CPU, reducing data retrieval latency. Compression algorithms often involve repeated access to the same data, they thus benefit from effective cache utilization.
- Cache Line Size and Prefetching: Optimizing cache line size and implementing effective prefetch techniques can ensure that the data required by the algorithm is readily available, minimizing stalls due to cache misses.
- Memory Bandwidth: High memory bandwidth allows rapid transfer of data between the CPU and memory sub-system. Compression algorithms that inherently process large amounts of data, can be bottlenecked by slow memory. High bandwidth ensures a steady flow of data, keeping the CPU busy and reducing idle times.

Optimizing cache architecture and ensuring sufficient memory bandwidth are critical for maintaining high data rates. This is particularly important in the case of LZ4 encoding. However, the available memory size may also impact throughput and result in processing delays.

HARDWARE SETUP

This performance assessment was realized with a QLS1046-Space development kit and a LX2160ARDB board from NXP, both of which have several interfaces available. The Linux operating system was used throughout our benchmarking. Notable aspects of the setup of the individual benchmark boards:

The QSL1046-Space evaluation board had 4GB of installed DDR4 memory rather than the larger 8GB option. In addition, the processor was running at 1.6GHz, instead of the possible 1.8GHz maximum clock.

The LX2160 processor has a decompression/compression acceleration engine (DCE) capable of handling data throughput rates up to 100Gbps. However, in the scope of this study this accelerator was not directly evaluated. Only the core pure computing performance was assessed. Obviously the DCE delivers significantly higher compression speed. Note also that our benchmark was only run at 2 GHz, instead of the 2.2GHz maximum clock. Note also that the findings are constrained by the available local DDR4 memory.

The simple benchmark objective is to target various types of processors, used in a variety of applications to have enough data to compare the compression performance.

The benchmark was executed on several processor platforms with multiple file types typical of those used in space applications. Some processors were directly tested by Teledyne e2v and for others we relied on previous testing as obtained in the bibliography [1].

Processors tested by Teledyne e2v

Processors	Tested Frequency (Max Frequency) - GHz	No. of cores tested (Max)	Architecture	Space qualified	Typical application
Texas Instruments DM3730	1	1	ARM	NO	Navigation
Amlogic S805	1.5	4	ARM	NO	Tv box
LS1046	1.6 (1.8)	4	ARM	YES	Space
Intel® Atom™ D525	1.8	2	x86	NO	Desktop
Intel® Core™ i7-2630QM	2	4	x86	NO	Mobile
LX2160	2 (2.2)	16	ARM	YES	Space
Intel® Core™ i5-1145G7	2.6	4	x86	NO	Desktop
Intel® Core™ i3-2105	3.1	2	x86	NO	Desktop
Intel® Core™ i5-2400	3.1	4	x86	NO	Desktop
Intel® Xeon® Processor E3-1225 v3	3.2	4	x86	NO	Server

Table 2: Processors tested

In this benchmark, several low-performance, radiation hardened processors were ignored because they would not be able to handle the compression task load. For example, the SAMRH71 from Microchip featuring an Arm Cortex M7 core delivering up to 200 DMIPS. That is 1000 times less throughput than the LX2160.

TEST FILE SCENARIOS

Several different file types were used for the compression benchmarking. The file types were considered representative of those typical of space applications. JPEG or RAW image data, which are typical image formats make interesting compression candidates as compression can accelerate image download times to close to real-time executional files can be deployed to enable 'over the air' software updates or facilitating the upgrade of AI plugins on a satellite system.

The characteristics of tested files are detailed below:

Name	Type	Description	Source	Size (MB)
x-ray	x-ray	X-ray medical picture	Hospital image[1]	8.5
dickens	.txt	Collected works of Charles Dickens	Project Gutenberg[1]	10.2
image	.raw	Picture without compression	Teledyne e2v	19.1
nci	database	Chemical database of structures	CACTVS[1]	33.55
mozilla	.exe	Tarred executables of Mozilla 1.0	Mozilla Project[1]	51.22
fireworks	.jpeg	A JPEG image	Snappy[1]	112

Table 3: File types used in this benchmarking process

Some observations concerning the file types listed. Firstly, the jpeg image is already compressed. Obviously, applying compression may be pointless, but it is still interesting to know whether LZ4 can provide further compression. A RAW image was tested to compare differences. In addition, text files represent an interesting test case because of the LZ4 algorithm structure which is built to have a library of words. Executable files are like text files because of a related code structure which might demonstrate considerable data redundancy. Finally, a database file (e.g .nci here) is interesting to evaluate because of the tabular structuring it typifies.

BENCHMARK

This benchmark is designed to compress files on individual processor cores, ensuring maximum performance and fast results. Below, there is a quick explanation of how it works:

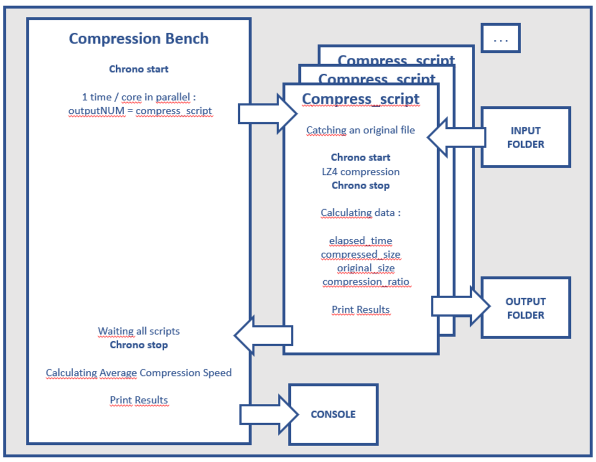

Figure 7: Block schematic of benchmark algorithm

Parallel processing: The benchmark simultaneously executes several compression scripts to compress files. This parallel processing significantly reduces the aggregate time required for file compression.

Performance metrics: The benchmark measures the compression encode time and calculates compression speed in MB/s. Moreover, the time taken for all threads is measured and a total compression speed in MB/s across all scripts is determined.

Continuous operation: The algorithm operates in an endless loop, enabling continuous file compression. The process is continually monitored.

COMPRESSION SPEED RESULTS

This benchmarking has focused on a single compression performance factor, namely the rate or speed of compression.
Computational speed is an important element in comparing processors.

The radar plot (fig. 8) shows interesting results. First, the LX2160 shows significant processing advantages compared with the Intel® Core™ i5-1145G7. Interestingly, note the similarity in the compression speeds (shape of plots) versus individual file types as marked on the chart spokes. The algorithm appears to deliver the same processing speed impact for every processor. This graphic therefore emphasizes the importance of algorithm selection. Moreover, the chart highlights a 4x speed advantage between the LX2160 and the slower LS1046 and Intel i5-1145G7.

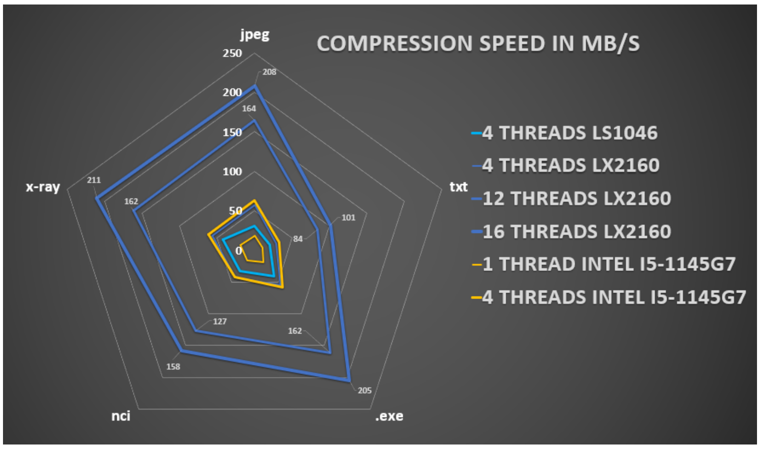

Figure 8: Compression speed on tested files with Teledyne e2v's processors

To allow comparison of Teledyne e2v's processors with an array of common commercial components, the results of a public benchmark were used. It is apparent that the public results only show single threading operation. For comparison fairness, all the public results were therefore multiplied by the number of available cores. However, recognize that this simple approach yields an untested approximation. This is justified since this benchmark also underestimates the compression capacity of the processors. However, it is important to note that certain file types, particularly the .nci file, that a multiplied one thread speed overestimates performance.

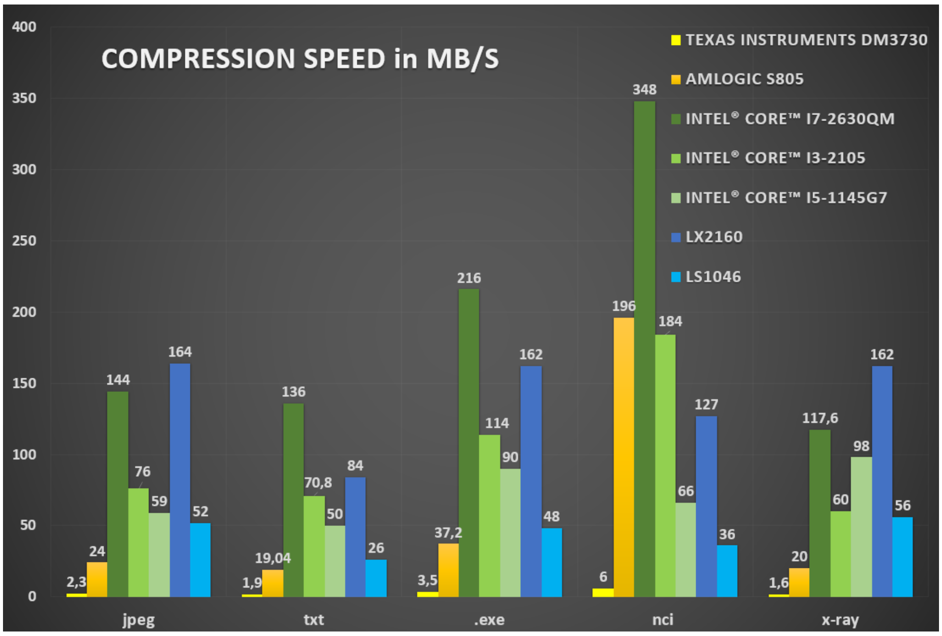

Figure 9: Compression speed on tested files with all processors

Figure 9 summarizes results. Five file types compressed using ten various processors. The LX2160 stands out here. On some file types, it leads the pack as demonstrated by the x-ray file results (162 MB/s). However, there are other file types where the Intel processors have an advantage e.g. the nci file type. Alas, this result should be treated with caution as it was not directly tested and is likely biased by the simplistic correction applied to extrapolate its speed.

CONCLUSION

This white paper discussed Teledyne e2v's space processors with impressive 64-bit compute power. Two types of compression, namely lossy and lossless have been considered. Some thought was given to specific compression algorithms and attention was given to the LZ4 algorithm as used in this benchmarking. This article also highlighted key processor performance factors important when considering compression methods.

This led to an attempt to benchmark compression of five file types using a series of ten processors, of which two, manufactured by Teledyne e2v are radiation tolerant space-grade components. The LS1046 performs on a par with commercial processors from Intel in terms of compression speed (MB/s). The newest LX2160 is four-times better than LS1046 in terms of compression power. It also has an impressive compression speed compared to the others. And recall that that benchmark was achieved without resorting to using the DCE - the on-chip decompression/compression acceleration engine; capable of data rates up to 100Gbps.

Thanks to the performance offered by this pair of space grade processors, it is possible to process and compress large datasets produced in flight; even with the data rate demands of the latest high resolution and extended sample rates of contemporary sensors. AI can also be used on-board [7], to extract data highlights before transmission to ground stations. This further reduces bandwidth requirements.

In future this analysis might be expanded to perform more detailed analysis on the commercial processors and test their multi-threading capabilities. Equally, it would be interesting to analyze the performance of FPGA compression, as applied lossy compression applications. Finally, it would be helpful to evaluate the compression rate per watt for all the commercial candidates.

SOURCES:

[1] Squash Compression Benchmark (quixdb.github.io)
[2] Lossless Compression: A Complete Guide | Adobe
[3] Lossy Compression: Everything You Need to Know | Adobe
[4] Lossless bit compression (article) | Khan Academy
[5] JPEG Compression Explained | Baeldung on Computer Science
[6] La compression de données - La compression de Lempel Ziv Welch (univ-mlv.fr)
[7] Deep-learning AI in Space enabled by Qormino® processing module | Teledyne e2v Semiconductors (teledyneimaging.com)
Executable Usage • Vitis Libraries • Reader • AMD Adaptive Computing Documentation Portal (xilinx.com)
https://ark.intel.com/content/www/fr/fr/ark/products
LZW (Lempel–Ziv–Welch) Compression technique - GeeksforGeeks

TELEDYNE e2v Semiconductors

For further information, please contact:
Manuel Blanco,
Applications Engineer,
Data Processing Solutions
Manuel.BLANCO@Teledyne.com

TELEDYNE e2v Semiconductors

For further information, please contact:
Vincent Thibault,
Applications Engineer,
Data Processing Solutions
Vincent.THIBAULT@Teledyne.com

TELEDYNE e2v Semiconductors

For further information, please contact:
Guillaume Nezot,
Applications Engineer,
Data Processing Solutions
Guillaume.Nezot@Teledyne.com

TELEDYNE e2v Semiconductors

For further information, please contact:
Thomas Guillemain,
Marketing & Business Development,
Data Processing Solutions
thomas.guillemain@teledyne.com