Tightly-coupled VS Loosely-coupled accelerators for data compression in space applications: a NOEL–V study case

Enrico Manfredi and Guido Masera
Politecnico di Torino, Italy
enrico.manfredi@studenti.polito.it – guido.masera@polito.it

Abstract. The growing complexity of space systems and missions is causing a rapid increase of onboard data size, to the point where efficient data compression algorithms become essential. The resource constrained environment requires careful consideration of how to optimize this procedure. On the other hand, NOEL–V is emerging as a promising space-grade processor, offering both high performance and reliability for critical space applications. Being built on the RISC–V ecosystem, NOEL–V inherits its ability to enhance performance through the integration of hardware accelerators with the processor. Tightly-coupled and loosely-coupled accelerators (TCAs and LCAs respectively) offer different benefits: the former are more compact but require more effort to integrate them, while the latter should be carefully designed to minimize their area. This work will compare those two approaches regarding the optimization of the CCSDS 121.0 algorithm, a lossless data compression technique recommended as a standard by the Consultative Committee for Space Data Systems. Synopsys ASIP Designer has been used to design and integrate the accelerators into a NOEL–V model. Two variants of TCAs will be proposed, each targeting specific bottlenecks of the algorithm, reaching a total clock cycle reduction of 85%, with an area increment of only 9% (based on 65 nm low power ASIC technology). The proposed memory-mapped LCA performs the entire compression procedure, reaching a speed-up factor of 480× and a 48% area increment with respect to the reference implementation. With a throughput of 7.82 Gb/s, the LCA turned out to be the far superior design in terms of Throughput-Area Ratio, but the TCAs proved to be a valid choice for lossless data compression optimization in area-constrained environments.

Keywords: Hardware design, accelerators, space, RISC–V, NOEL–V, data compression, CCSDS 121.0, Asip Designer

1 Introduction

Modern satellite missions and space exploration platforms face a critical challenge when it comes to efficient data processing. The increasing complexity of payloads and Earth observation applications has led to a noticeable size increment of data generated onboard. For this reason, the implementation of advanced data compression techniques has become essential for both data storing and transmission. Although a software version of the compression algorithm is the most portable solution, it is also not efficient, while a fully-dedicated hardware implementation lacks in flexibility, requiring to redesign the hardware every time a change is needed. ASIPs can be a good trade-off to greatly increase the performance thanks to hardware accelerators without compromising too much area and flexibility. This work will compare tightly-coupled and loosely-coupled accelerators, enhancing the performance of the recommended **CCSDS 121.0** standard for lossless data compression in space applications.

2 Background: NOEL–V, CCSDS 121.0 and ASIP Designer

RISC–V is a modern open instruction set architecture (ISA) that fosters processor innovation and enables open-source hardware development with the integration of domain-specific ISA extensions, accelerators, and coprocessors. The RISC–V that has been chosen for this study is the NOEL–V [1], a new candidate in the space industry that proved to be a suitable alternative for novel space-graded processor architectures. The NOEL–V chosen for this work is the General Purpose one, which features a 64-bit 7-stage pipeline, dual-issue, and FPU. The chosen algorithm for this work is the **CCSDS 121.0** standard, specifically the B-3 version [2], regarding lossless data compression. It consists of two separate functional parts: the preprocessor and the Adaptive Entropy Coder. The function of the preprocessor is to decorrelate input data and reformat them into non-negative integers, applying a reversible function to each block of input samples. The preprocessor adopted for this work is the **unit delay predictor** with the **prediction error mapper** described in the standard. The Adaptive Entropy Coder applies concurrently several compression algorithms to a block of consecutive preprocessed samples, and the option that yields the shortest encoded length is selected for transmission. A unique bit sequence identifier is also attached to the code block to indicate to the decoder which decoding option to use. The code implementation of the algorithm used for this work can be found in the OBPMark repository [3], an open source set of computational performance benchmarks developed specifically for spacecraft on-board data processing applications.
To facilitate the development of the accelerators, **Synopsys ASIP Designer** [4] has been used. With this tool, processor models can be seamlessly customized and simulated, and its integrated profiling capabilities allows to easily identify bottlenecks in the computation. The *trv64p5* ASIP Designer model, a 64-bit RISC–V processor

with 5-stage pipeline included in the tool, has been modified to emulate the architecture of the NOEL–V. After adding the two additional pipeline stages, the FPU and the dual-issue with all the correct pairing constraints, the resulting processor model is a close match to the NOEL–V. Some more complex mechanisms pertaining to the original core have not been included in the ASIP Designer model due to some limitations of the tool, but the final result has all the important features to develop the accelerators.

3 Tightly-coupled accelerators

Tightly coupled accelerators (TCAs) resides within the CPU's microarchitecture, necessitating core and toolchain modifications. They usually need instruction set extension, meaning that the ISA of the processor is extended in order to use the accelerator. The first profiling of the code reveals that the majority of the clock cycles are spent in preliminary and standard library functions like malloc and calloc. A small part of the code has been rewritten to reduce their usage, shortening the simulation time significantly. The subsequent profiling highlighted the functions to be optimized.
In the original algorithm, compressed data was stored one bit at a time. The **writeWord** TCA, shown in Figure 1, enhances this procedure thanks to an internal buffer, where the compressed data is appended multiple bytes at a time. The TCA then writes the 32-bit words in the memory as a standard store operation. The two **J-block** TCAs depicted in Figure 1 enhance the identification of the best compression scheme. The first one improves the compressed block size calculation, holding the intermediate sum value in an internal register and also storing the processed sample directly in the memory location used before the actual compression. The other J-block TCA is used to enhance the calculation of the H-sample starting from two input samples. Another variant of these TCAs where double the samples are computed every clock cycle has also been tested.
Finally, the **preprocessor** TCA has been developed. This is the most straightforward accelerator, as it is a simple hardware implementation of the preprocessor described in the standard. Since this step is optional for certain algorithms, the entire preprocessor can be disabled by writing a flag in an internal register.
To prove that the TCAs work even outside their ASIP Designer model, they were also integrated and tested inside the NOEL–V core within GRLIB. This process required to modify the processor core, as well as the NCC compiler [5], to properly use the new custom instructions. The obtained NOEL–V ASIP retains all the advanced features of the original processor and enhances the data compression using the TCAs developed with ASIP Designer.

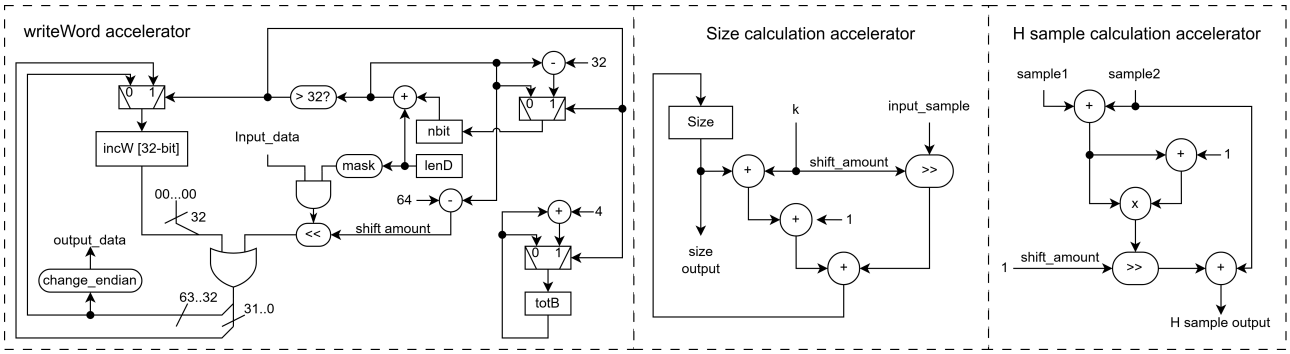

Fig. 1. writeWord and J-block TCAs simplified schematics

4 Loosely-coupled accelerator

Unlike TCAs, loosely-coupled accelerators (LCAs) are often memory-mapped and connected via system buses to handle compute-heavy tasks with large data sets. The developed LCA features an AHB slave, used by the processor to operate the accelerator, and AHB master, used to read input samples directly from the data memory. To maximize the throughput, a standard memory interface is used to store the compressed samples, allowing concurrent read and write operations. The ASIP Designer memory model has therefore been updated, using as a reference the AHBDPRAM found in the GRLIB IP library [6], a dual port SRAM with both an AHB interface and a back-end memory port. Configuration parameters are sent over using the AHB slave interface, and also commands are given in the same way. Pipelining is used in order to maintain small combinational paths, and the sizes of the various compression techniques are calculated in parallel, as shown in Figure 2. A known limitation of the proposed implementation is that it stops receiving input samples until the first phase of the Sample Splitting encoding is completed, reducing the throughput when using this technique. This is done to prevent the samples stored inside the LCA from being overwritten prematurely. For that matter, the profiling of all the proposed architectures was performed under the worst-case scenario, where all blocks are encoded using the Sample Splitting option.

To test the accelerator with a real processor, the LCA was also integrated in a NOEL–V subsystem. After creating a suitable simulation model for the AHBDPRAM memory, the designed accelerator was attached to both the AHB bus and the back-end memory port and thoroughly tested.

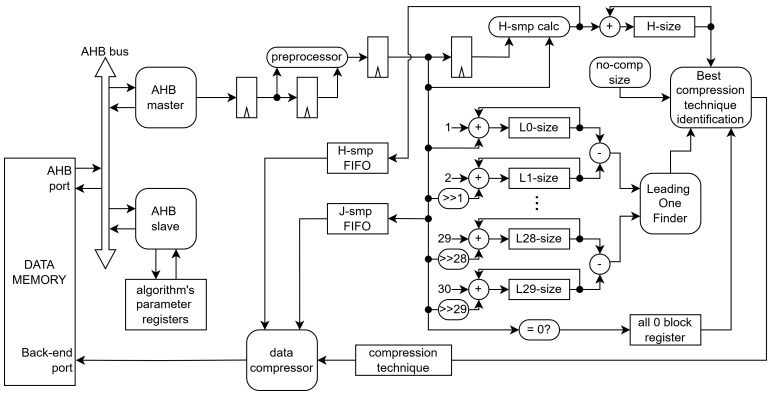

Fig. 2. LCA simplified schematic

5 Results and Comparisons

The code taken from OPBMark was first profiled on the unoptimized ASIP Designer model, which was also synthesized into a 65 nm low-power ASIC using Synopsys Design Compiler. ASICs provide higher performance, lower power consumption, and greater reliability, crucial for constrained environments like space systems, and are particularly preferred for tightly coupled accelerators. The optimized ASIP Designer models were subsequently profiled and synthesized with the same technology. The critical path does not belong to any of the accelerators, allowing all the designs to be synthesized with the same minimum clock period of the base design, i.e. 2 ns. The throughput is calculated with the equation $TH = computed\ bits \cdot \frac{f_{max}}{CC}$, where CC corresponds to the total number of clock cycles needed to compute all the input samples in the application code, and f_{max} is the maximum model frquency of 500 MHz. Table 1 compares the results obtained for the reference implementation with each optimized design. For the TCAs, the A version reaches an 85% clock cycle reduction, meaning a $\times$6.7 speed up with respect to the reference implementation. The B version computes two samples with every J-block instruction, meaning that the performances are increased even further, totalling to a $\times$9.6 speed-up factor, with a 14% area increment. From the Throughput-Area Ratio (TAR) metric, it is possible to observe that the B variant is slightly advantageous with respect to the other one, if the application allows for the extra area investment. The loosely-coupled version, on the other hand, gives the best performance increment out of all designs, decreasing the clock cycles by 99.8%, which corresponds to a $\times$480.1 speed-up factor. The achieved throughput of 7.82 Gb/s represents the worst-case scenario where all the blocks are encoded using the sample splitting option. By removing this limitation the throughput could be even higher, matching the value of 21.35 Gb/s already achieved when using only the other techniques. Either way, the TAR strongly favours the LCA, which is undoubtedly the superior choice if the 47.5% area increment is accepted. At the time of writing, no works similar to the proposed TCAs have been reported. Project [7] obtained comparable performance results, but the approach used is completely different, featuring a multicore and GPU implementation of the algorithm. On the other hand, many implementations similar to the proposed LCA can be found in the literature, even though the only reference to an ASIC implementation is found in [8]. It is worth noting that, in both the proposed design and [8], small memories like the internal FIFOs have been mapped to FFs, causing an area overhead. These results should then be considered as upper bounds, because synthesis with appropriate memory macros should yield a lower area.

Metrics	TCA variant A: 1 sample/inst (step-by-step)					TCA variant B: 2 samples/inst	Ref. LCA (added AHB)	LCA with core	LCA only	SHyLoC of [8] (Set 3 w mem)
	Reference	rewriting C	writeWord	J -block	Preprocessor					
CC	1,029,520,644	576,095,801	283,249,960	203,407,562	153,878,674	107,093,829	1,029,520,644	2,144,284	-	-
CC reduction VS Reference	0.0%	44.0%	72.5%	80.2%	85.1%	89.6%	0.0%	99.8%	-	-
Speed-up VS Reference	1.00	1.79	3.63	5.06	6.69	9.61	1.00	480.12	-	-
CC reduction VS rewriting C	-	0.0%	50.8%	64.7%	73.3%	81.4%	-	99.6%	-	-
Speed-up VS rewriting C	-	1.00	2.03	2.83	3.74	5.38	-	268.67	-	-
ASIC synthesis technology	l.p. 65 nm	l.p. 65 nm	l.p. 65 nm	l.p. 65 nm	l.p. 65 nm	l.p. 65 nm	l.p. 65 nm	l.p. 65 nm	l.p. 65 nm	DARE 180 nm
Max Frequency [MHz]	500	500	500	500	500	500	500	500	709	194.17
Total Core Area [μm^2]	168,478	168,478	171,657	181,103	184,244	192,104	175,138	258,310	94,532	2,550,000
Area Increment	0%	0%	1.9%	7.5%	9.4%	14.0%	-	47.5%	-	-
Throughput at f_{max} [Gb/s]	0.02	0.03	0.06	0.08	0.11	0.16	0.02	7.82	11.09	6.21
TAR	96.73	172.86	345.06	455.44	591.76	815.49	93.05	30289.80	-	-

Table 1. ASIC design results and comparisons

The proposed LCA has also been implemented on the XCVU3P FPGA for an easier comparison with other works. Due to the aforementioned limitation, the worst-case performance of this work are slightly lower than the

one reported for work [9], despite the higher frequency of the proposed design. Solving this issue would lead to better results, with additional improvements possible by reducing the critical path. Paper [10] parallelize part of the computation of SHyLoC, achieving a throughput higher than the proposed solution, at the cost of higher resource utilization. The work in [11] presents an efficient implementation of the algorithm, but detailed material is hard to find, and implementation specifics are not provided in the available sources. The solution adopted for work [12] is constrained to image data compression targeting remote sensing applications. This means that the maximum bit depth for the input samples is 16 bits, whereas the proposed is 32 bits. The architecture is then tailored around this constraint, achieving a smaller area footprint and better performance than this work.

Metrics	Proposed LCA only	SHyLoC of [9] (D = 32)	Parallel121 of [10]	USES-32 of [11]	Work of [12]
FPGA	XCVU3P	XQR5VFX130	XCKU040	EP3SE50F484C2	XC6VLX75T
Max Frequency [MHz]	143	79.9	121.5	-	313
LUTs / DSPs	8929 / 3	7670 / 5	28329 / 4	6255 / -	9% slices
FFs / BRAM	4275 / 96 (LUTRAM)	2291 / 0	8774 / 0	3383 / ~16 Kb	5% BRAMs
Throughput at f_{max} [Gb/s]	2.24	2.56	7.78	6.40	4.67

Table 2. FPGA design results and comparisons

6 Conclusion

This work greatly enhanced the performance of the CCSDS 121.0 algorithm, exploring various strategies in the solution space. Given the widespread support of the RISC–V project from various vendors and corporates, the NOEL–V proved to be an optimal candidate processor for this optimization, and ASIP Designer allowed for rapid and intuitive development of the accelerators. The LCA proved to be the best design for what concerns TAR, reaching a throughput of 7.82 Gb/s with an area increment of 47.5%. On the other hand, TCAs still increased the performance noticeably while remaining compact enough for area constrained devices. All the designed TCAs and the LCA have also been integrated and tested inside the NOEL–V system, and prooved to work correctly in every operating condition. Future work will be done to remove the aforementioned limitation of the Sample Splitting encoding on the LCA, matching the rate of one sample per clock cycle already achieved by the other compression techniques. Other possibilities concern improving the reference C code implementation of the OBPMark repository, making it more efficient, and producing better results for what concerns TCAs performance. Finally, this comparison can be extended to the CCSDS 123.0 multispectral and hyperspectral image compression standard, especially exploring multicore and parallel implementations.

References

[1] Jan Andersson. “Development of a NOEL-V RISC-V SoC Targeting Space Applications”. In: *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. 2020, pp. 66–67. DOI: 10.1109/DSN-W50199.2020.00020.

[2] Consultative Committee for Space Data Systems (CCSDS). *CCSDS 121.0 Lossless Data Compression*. Tech. rep. CCSDS 121.0-B-3. Blue Book – Issue 3. CCSDS, Nov. 2022. URL: https://public.ccsds.org/Pubs/121x0b3.pdf.

[3] OBPMark. *On-Board Processing Benchmark Suite*. 2024. URL: https://obpmark.github.io/.

[4] Synopsys, Inc. *ASIP Designer – Application-Specific Instruction-Set Processor Development Tool*. 2024. URL: https://www.synopsys.com/dw/ipdir.php?ds=asip-designer.

[5] Cobham Gaisler. *NOEL-V Bare-Metal Cross Compiler*. 2025. URL: https://www.gaisler.com/products/noel-bare-metal-cross-compiler.

[6] Cobham Gaisler. *GRLIB IP Library User’s Manual*. 2023. URL: https://download.gaisler.com/products/GRLIB/doc/grip.pdf.

[7] Alvaro Jover-Alvarez et al. “Space Compression Algorithms Acceleration on Embedded Multi-core and GPU Platforms”. In: *Ada Lett.* 42.1 (Dec. 2022), pp. 100–104. ISSN: 1094-3641. DOI: 10.1145/3577949.3577969. URL: https://doi.org/10.1145/3577949.3577969.

[8] European Space Agency (ESA). *SHyLoC Datasheet v1.0*. 2021. URL: https://amstel.estec.esa.int/tecedm/ipcores/SHyLoC_Datasheet_v1.0.pdf.

[9] Yubal Barrios et al. “SHyLoC 2.0: A Versatile Hardware Solution for On-Board Data and Hyperspectral Image Compression on Future Space Missions”. In: *IEEE Access* 8 (2020), pp. 54269–54287. DOI: 10.1109/ACCESS.2020.2980767.

[10] Samuel Torres-Fau et al. “CCSDS121-based High-Performance Hardware Architecture for Real-Time Data Compression”. In: *2023 European Data Handling Data Processing Conference (EDHPC)*. 2023, pp. 1–8. DOI: 10.23919/EDHPC59100.2023.10395929.

[11] L. Miles et al. “Over 3 Gb/s Universal Lossless Compressor for Space Use”. In: *Proceedings of the ReSpace/MAPLD 2011 Conference*. New Mexico, USA, 2011.

[12] Nektarios Kranitis et al. “Efficient field-programmable gate array implementation of CCSDS 121.0-B-2 lossless data compression algorithm for image compression”. In: *Journal of Applied Remote Sensing* 9 (May 2015), p. 097499. DOI: 10.1117/1.JRS.9.097499.