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

Enrico Manfredi

Prof. Guido Masera

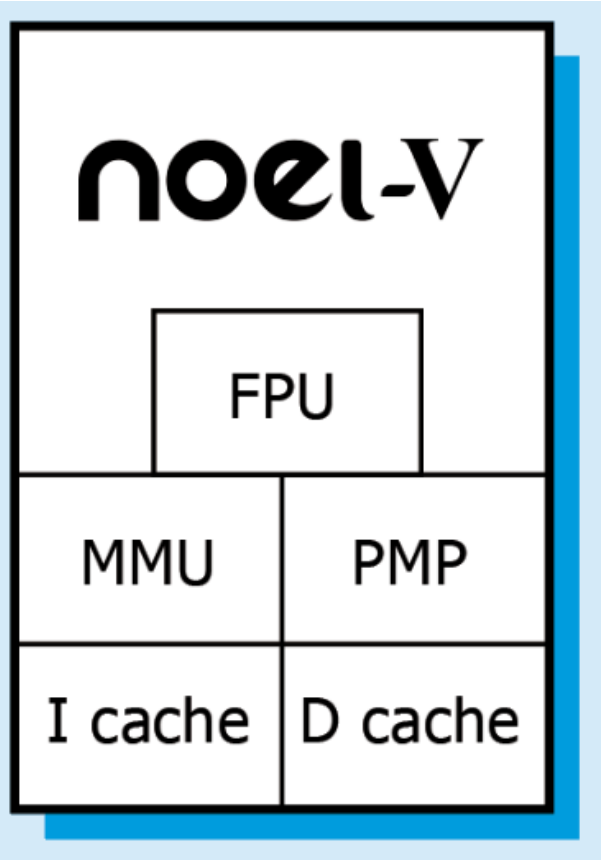
Politecnico di Torino

DET

RISC-V in space Workshop – April 2025

OBJECTIVES

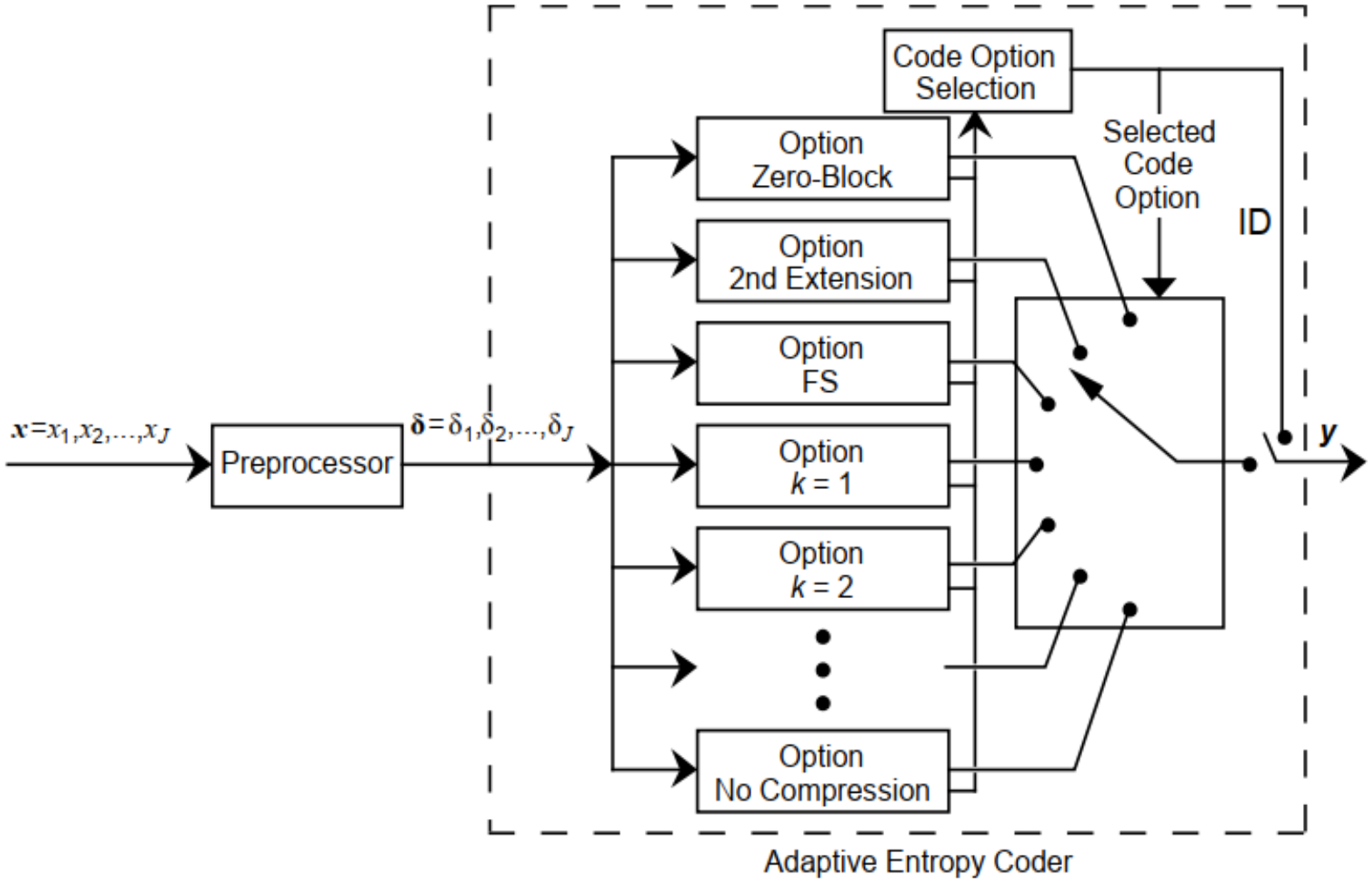
- Acceleration of the CCSDS 121.0 algorithm using ASIP Designer, realizing two different kinds of accelerators:
  - Tightly-coupled
  - Loosely-coupled
- Simulation, synthesis and results comparison of the two approaches.
- Integration of the accelerators in the HDL processor from the official library.



THE CCSDS 121.0 ALGORITHM

The algorithm chosen for the optimization is the **CCSDS 121.0-B-3** data compression standard.

The reference code implementation is the **OBPMark** GitHub repository, an open-source set of spacecraft on-board processing benchmarks.



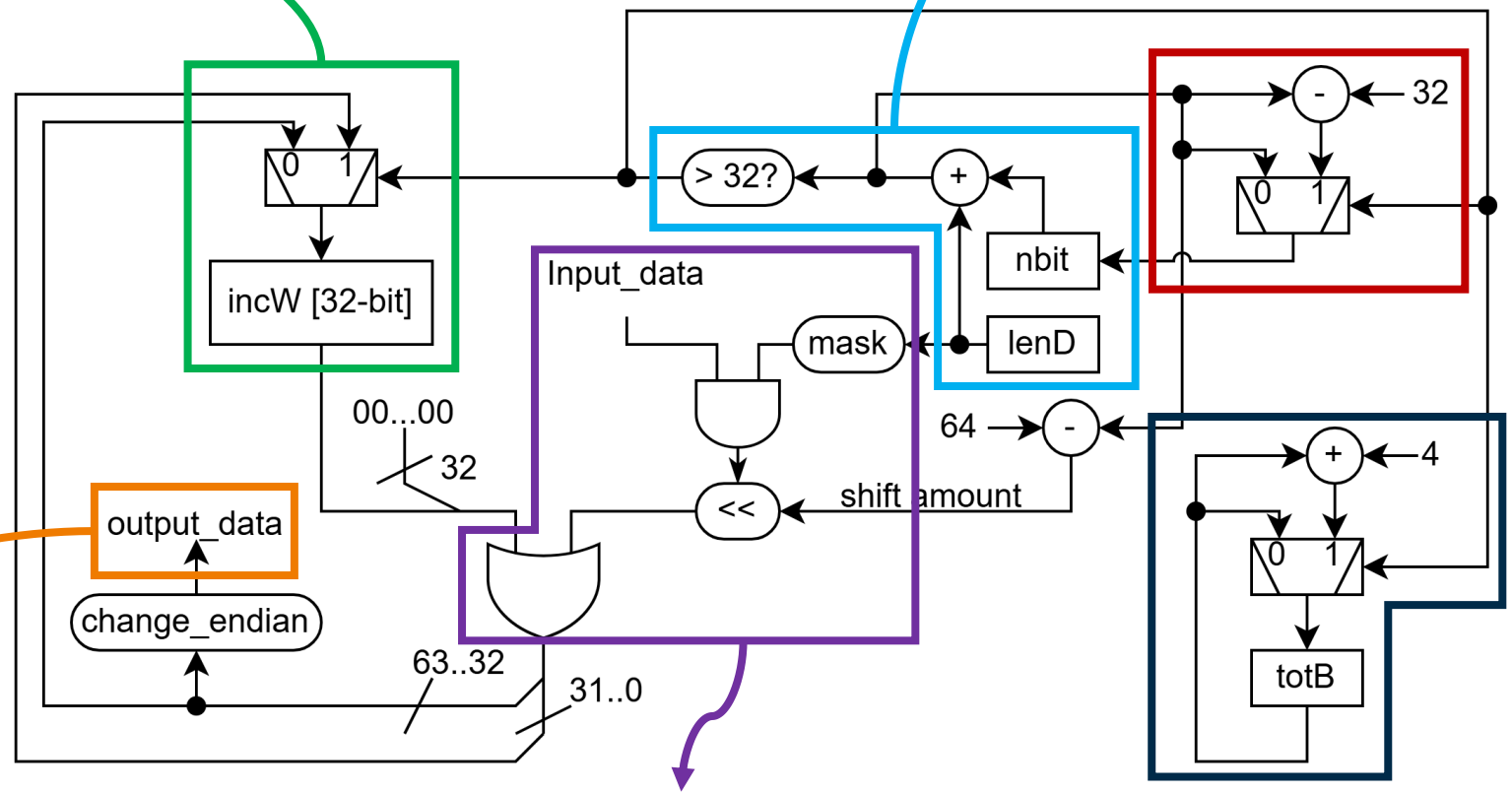
ALGORITHM BOTTLENECKS

- **Storing of compressed data**: compressed samples are stored into memory one bit at a time.
- **J-block compressed size calculation**: this is the most time-consuming step of the compression technique identification. A sample block is read several times to evaluate the size achieved by each compression scheme.
- **Data preprocessing**: if active, the optional preprocessing stage is performed on each input sample. It consists mainly in simple confrontations and arithmetic operations.

WRITEWORD TCA

Incomplete 32-bit word with the valid bits to write in the memory. It is updated with the 32 MSBs if the number of valid bits is less than 32, and with the 32 LSBs otherwise

Check if valid bits in the buffer are more than 32



Update the number of valid bits in the incomplete word

Number of bytes written in memory through the accelerator

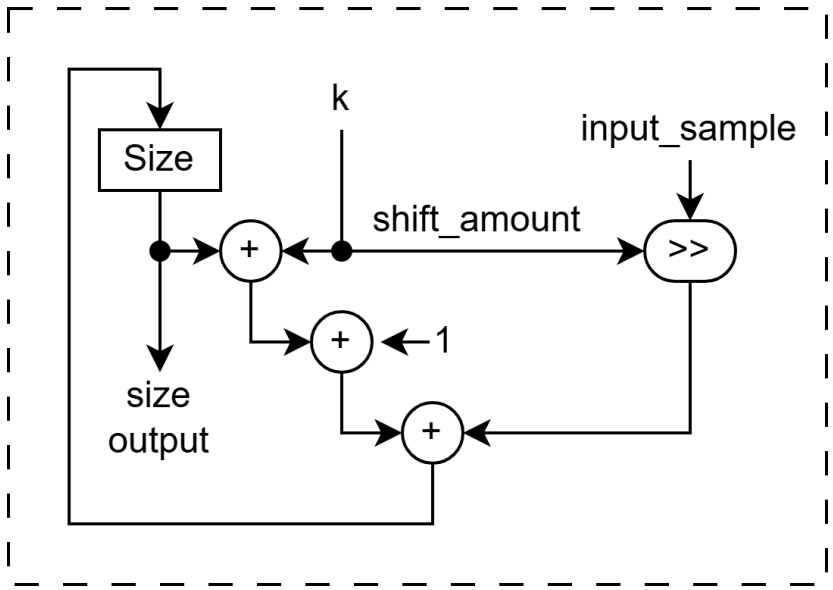
32-bit word to write in memory

Input data is masked to eliminate unwanted bits, left-shifted to append the new data in the right locations and then OR-ed to the current incomplete word extended on 64 bits

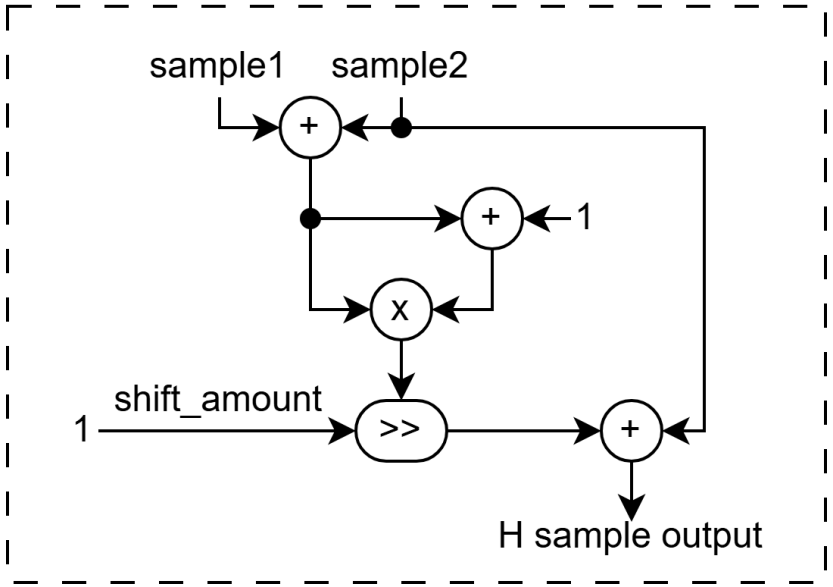
PREPROCESSOR AND J-BLOCK SIZE TCAs

The developed **preprocessor** is a faithful implementation of the one defined in the standard, featuring a **unit delay predictor** and a **prediction error mapper** that allow to preprocess an input sample in a single clock cycle.

The two following **J-block** accelerators have also been realized in a second version where **twice the number of input samples** are analysed simultaneously.



Size calculation accelerator



H sample calculation accelerator

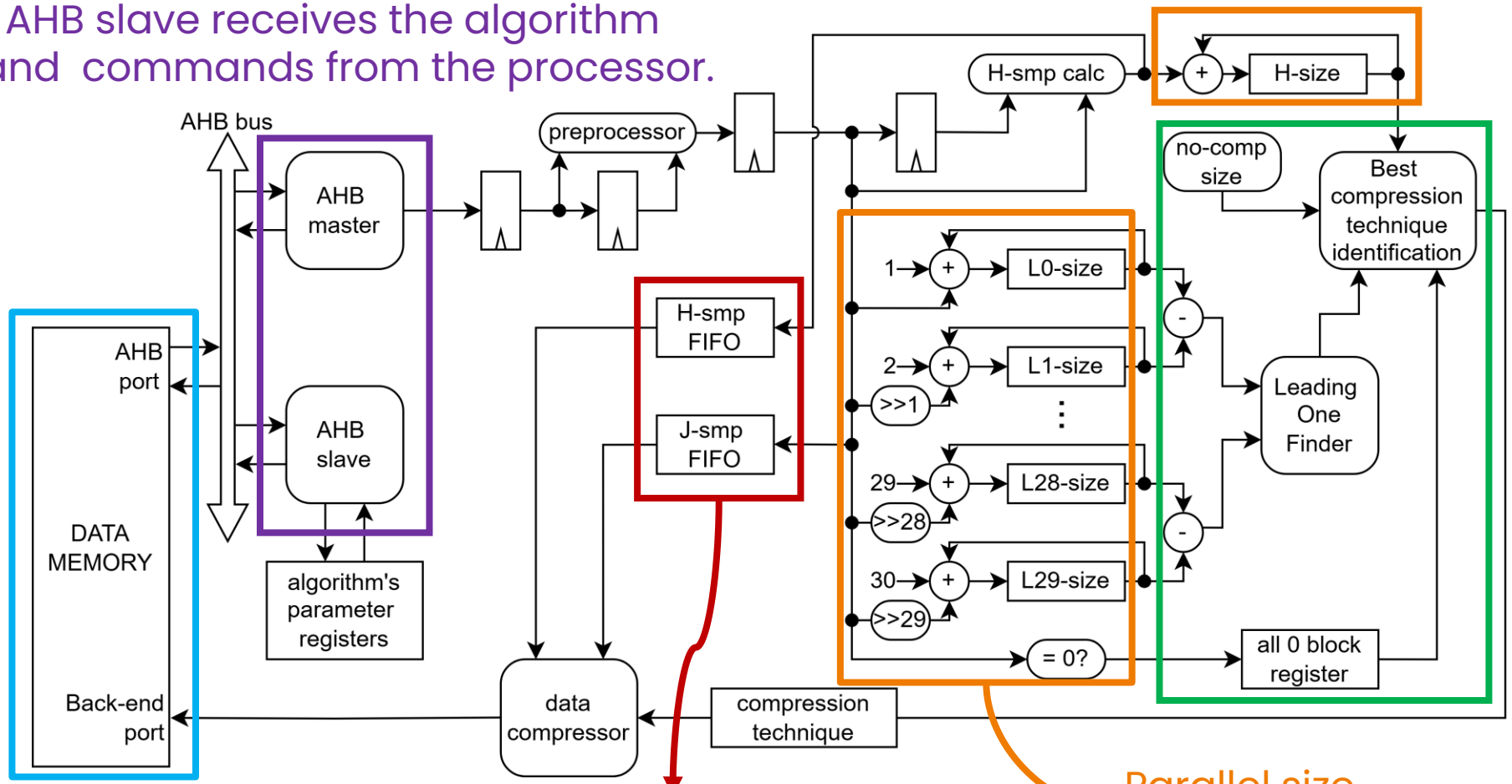
TCA RESULTS

If the system supports 64-bit operations, the B variant is slightly advantageous, as shown by the throughput to area ratio (TAR) metric:

Metrics	TCA variant A: 1 sample/inst (step-by-step)					TCA variant B: 2 samples/inst
	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
CC reduction VS Reference	0.0%	44.0%	72.5%	80.2%	**85.1%**	**89.6%**
Speed-up VS Reference	1.00	1.79	3.63	5.06	6.69	9.61
CC reduction VS rewriting C	-	0	50.8%	64.7%	73.3%	81.4%
Speed-up VS rewriting C	-	1	2.03	2.83	3.74	5.38
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
Max Frequency [MHz]	500	500	500	500	**500**	**500**
Total Core Area [μm^2]	168,478	168,478	171,657	181,103	184,244	192,104
Area Increment	0%	0%	1.9%	7.5%	**9.4%**	**14.0%**
Throughput at f_{max} [Gb/s]	0.02	0.03	0.06	0.08	0.11	0.16
TAR	96.73	172.86	345.06	455.44	**591.76**	**815.49**

LOOSELY-COUPLED ACCELERATOR

The AHB master reads the samples from the data memory. The AHB slave receives the algorithm parameters and commands from the processor.



The dual port memory allows concurrent read and write operations

The best compression technique is found in one clock cycle and saved for the memory write phase

FIFOs hold the J and H samples of a block until the writing process is completed

Parallel size calculation

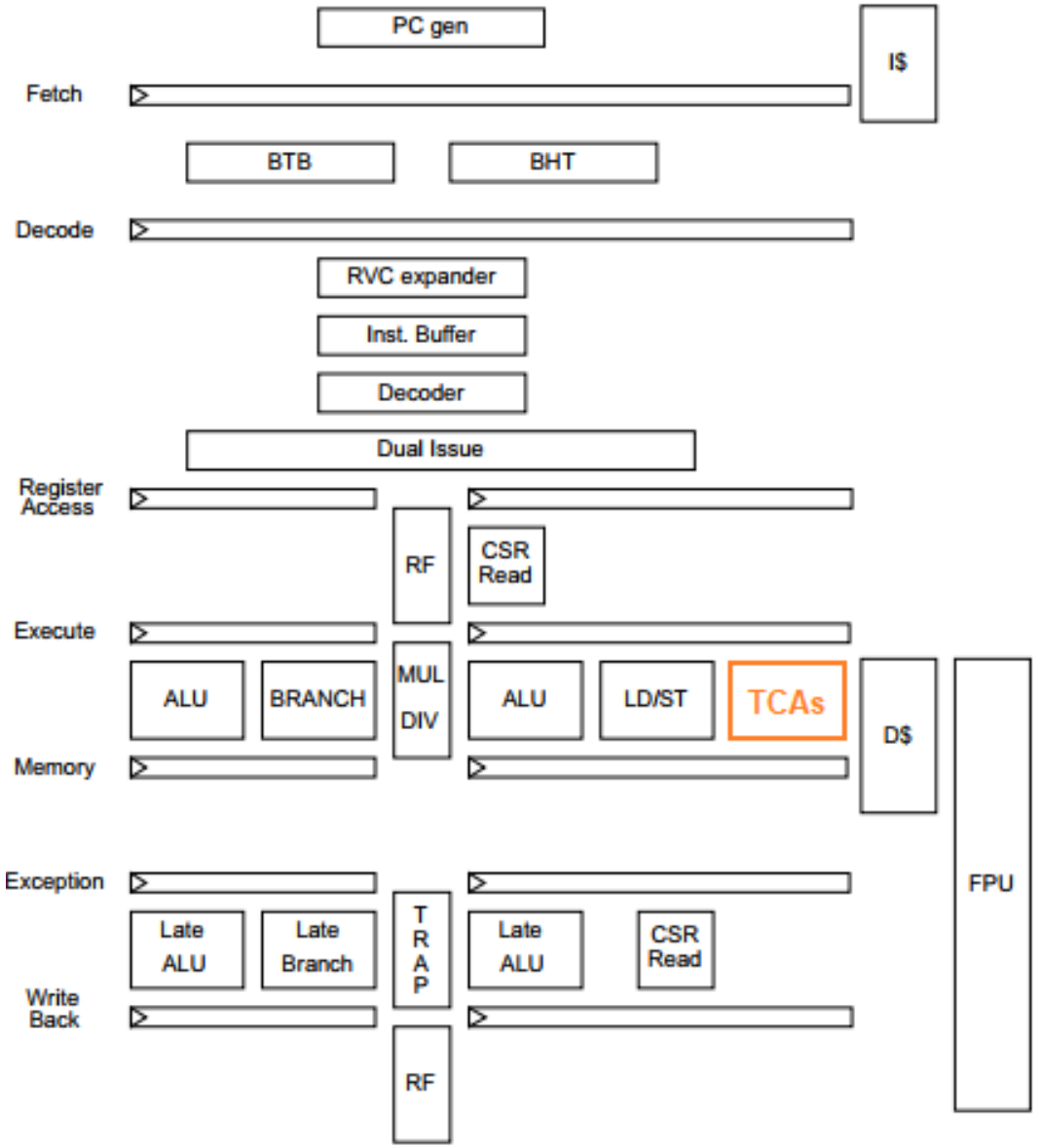
LCA RESULTS

Metrics	Ref. LCA (added AHB)	LCA
CC	1,029,520,644	2,144,284
CC reduction	0	99.8%
Speed-up	1.0	**480.1**
ASIC synthesis technology	l.p. 65 nm	l.p. 65 nm
Max Frequency [MHz]	500	500
Total Core Area [μm^2]	175138	258310
Area Increment	0%	**47.5%**
Throughput at f_{max}[Gb/s]	0.016	7.824
TAR	93.05	**30289.80**

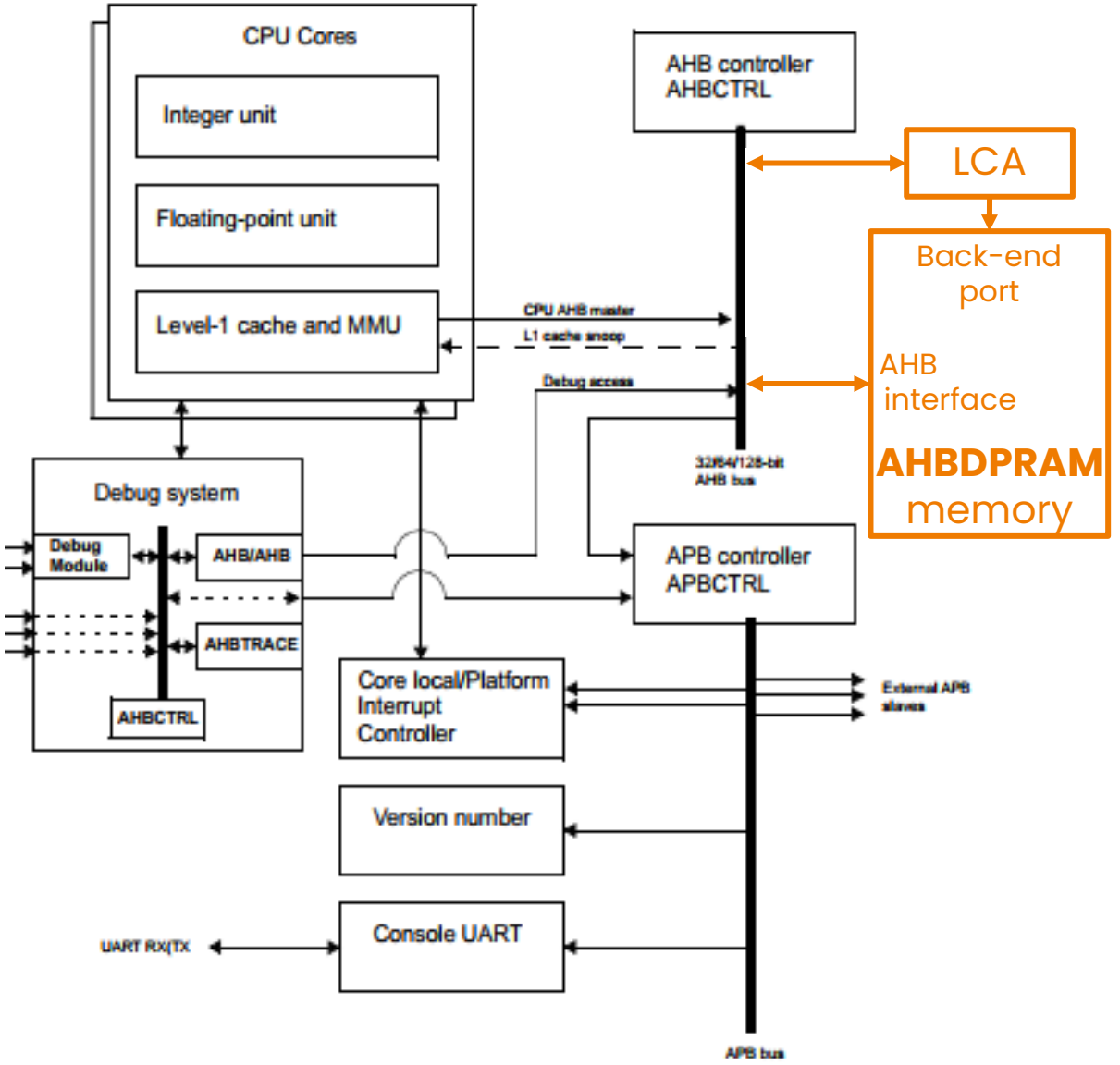
Side notes:

- The reference area is different from the TCA's one because the AHB interface has been added to the processor.
- FIFOs have been synthetised with flip-flops, therefore synthesis with appropriate memory macros should yield lower area values.

TCA INTEGRATION



LCA INTEGRATION



TCA VS LCA COMPARISON

Metrics	TCA variant A: 1 sample/inst	TCA variant B: 2 samples/inst	LCA
CC reduction	85.1%	89.6%	99.8%
Speed-up	6.7	9.6	480.1
Max Frequency [MHz]	500	500	500
Area Increment	9.4%	14.0%	47.5%
Throughput at f_{max} [Gb/s]	0.109	0.157	7.824
TAR	591.76	815.49	30289.80

TCAs pros and cons:

+ Low area increment for good performance.
+ Software can handle algorithm changes.
+ Can be used selectively.
- Require more effort for the integration.
- Perform better with proper coding style.
- Best if implemented in ASIC technology to ensure short processor connections.

LCAs pros and cons:

+ Best performance increment.
+ More portable and low integration effort.
+ Can work concurrently with the processor.
- High area increment.
- An ASIC implementation ensures maximum performance, but an FPGA implementation is more flexible.

COMPARISON WITH OTHER WORKS

At the time of writing no other works like the proposed TCAs have been found.

On the other hand, some works similar to the proposed LCA are available. To facilitate the comparison, the developed LCA alone has also been implemented on the XCVU3P FPGA.

Metrics	Proposed LCA only	SHyLoC of [1] (D = 32)	Parallel121 of [2]	USES-32 of [3]	Work of [4]
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

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

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

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

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

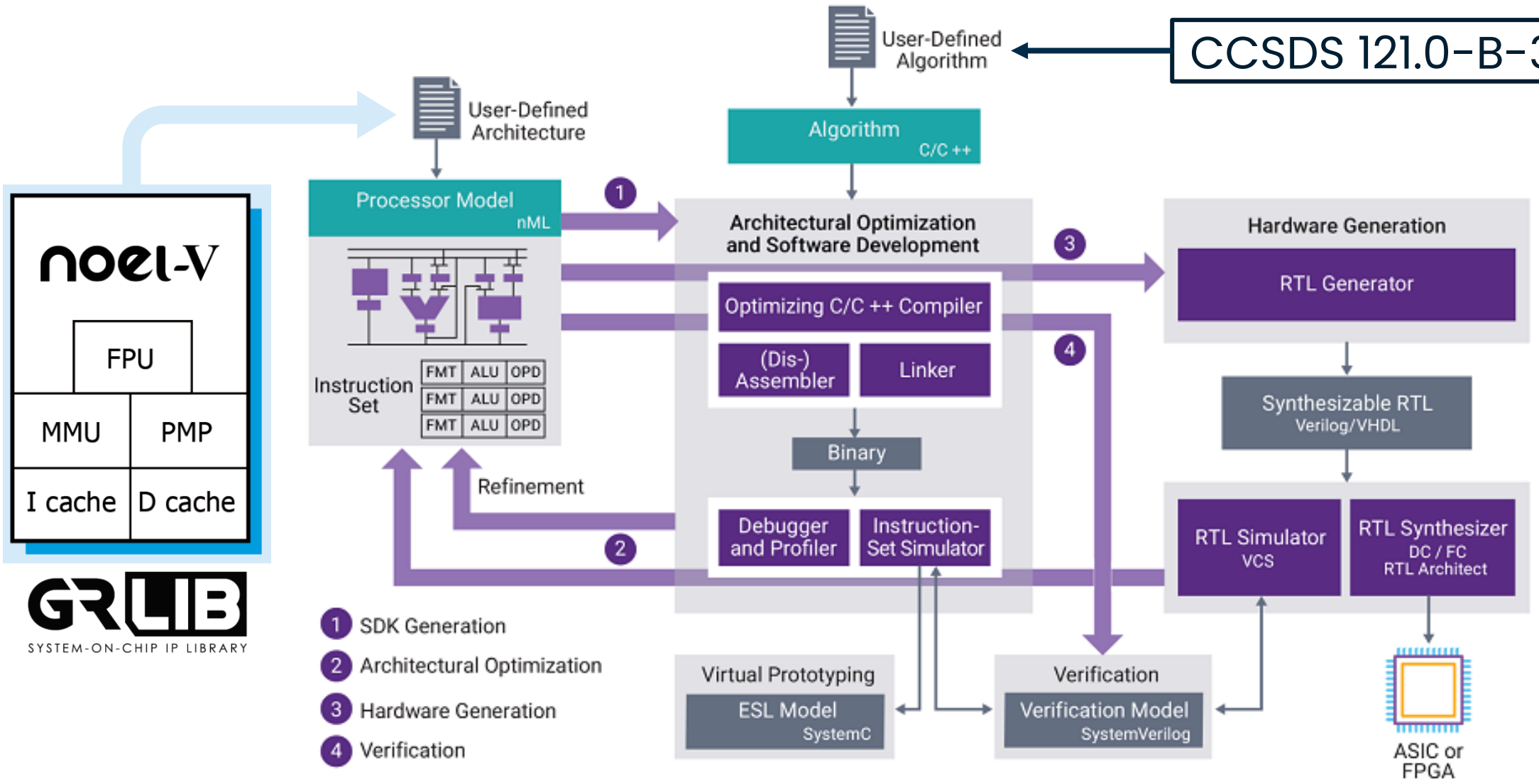
Thank you for your attention

Enrico Manfredi - enrico.manfredi@studenti.polito.it
Prof. Guido Masera - guido.masera@polito.it

Politecnico di Torino

DET

BACKUP 1: PORTING INTO ASIP DESIGNER



BACKUP 2: REWRITING THE CODE

The first profiling of the application done with ASIP Designer highlights the following functions as the most computationally intensive:

Function	Calls	Cycles Tot (func)	Cycles Tot (% func)	Cycles Tot (func + desc)	Cycles Tot (% func + desc)
memcpy	164415	253080042	24.58%	253080042	24.58%
memset	148307	217029535	21.08%	217029535	21.08%
writeWord	1029312	195455024	18.99%	195455024	18.99%
GetSizeSampleSplitting	209079	94712787	9.20%	94712787	9.20%
writeValue	2059226	72874215	7.08%	72874215	7.08%
preprocess_data	4	61018874	5.93%	101110574	9.82%

The starting code has been slightly rewritten to reduce the use of memcpy and memset, shortening the simulation time and highlighting the real computational bottlenecks.