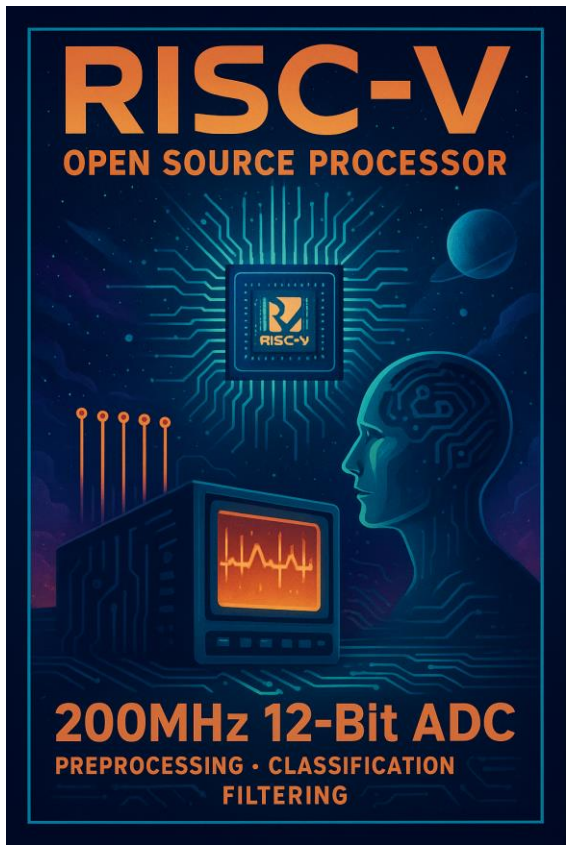
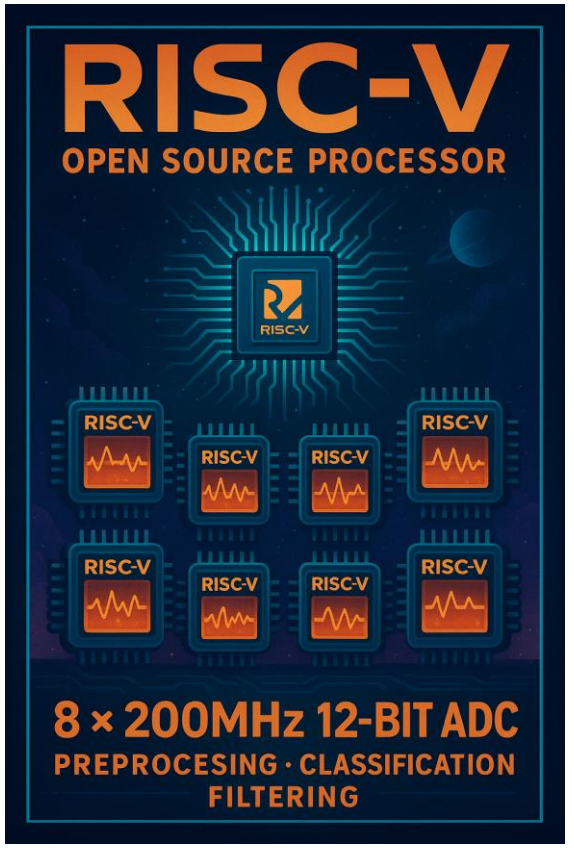
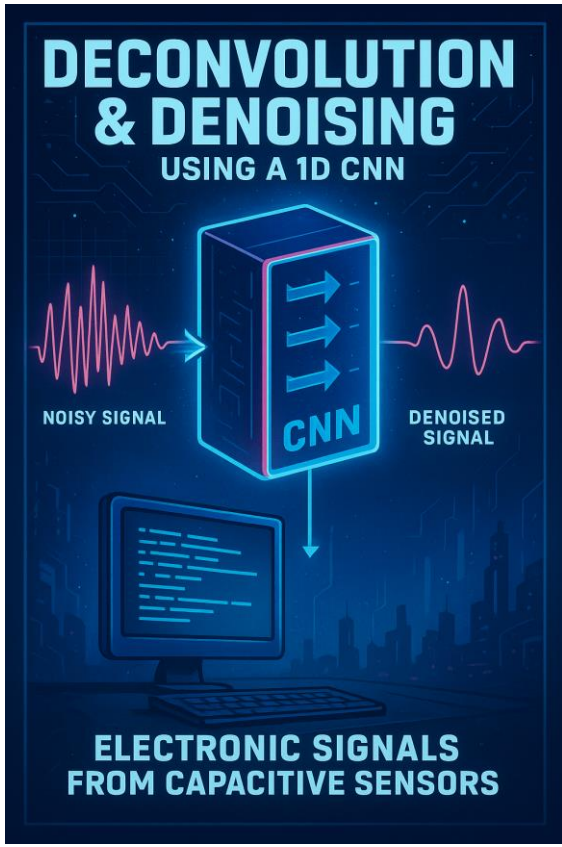
– RISC-V Multi-Parallel-Processors for Data Acquisition–







Frederic Druillole THINK – EL2I

Test of Hardware Inferring Neural networK

Reseau DAQ – Hardware-Firmware-Software-

Phase 2

5 phenomena which perturbate the front-end measurement:

- **Pile-Up**
- **Balistic Error**
- **Cross-talk**
- **Intrinsic detector noise**
- **Electronic channel noise**

$$\sigma^2_{total} = \sigma^2_{det} + \sigma^2_{pu} + \sigma^2_{ba} + \sigma^2_{elec}$$

→ Optimum Problem : Minimisation

Optimal filtering channel

Elts of Interest

Charge conversion

Charge mobility

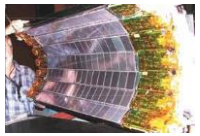
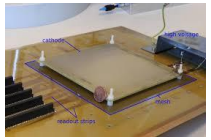
Signal amplifier

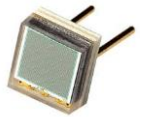
Signal optimum

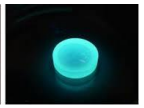
Measures

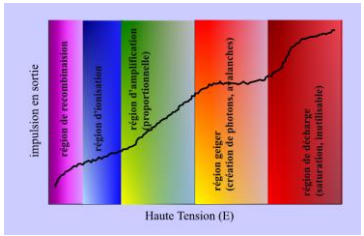
Photons (gamma, X, light)
Charged particles(electrons, muons, ions)
Neutral Particles (neutron, neutrino, alpha)

Technologies of detection







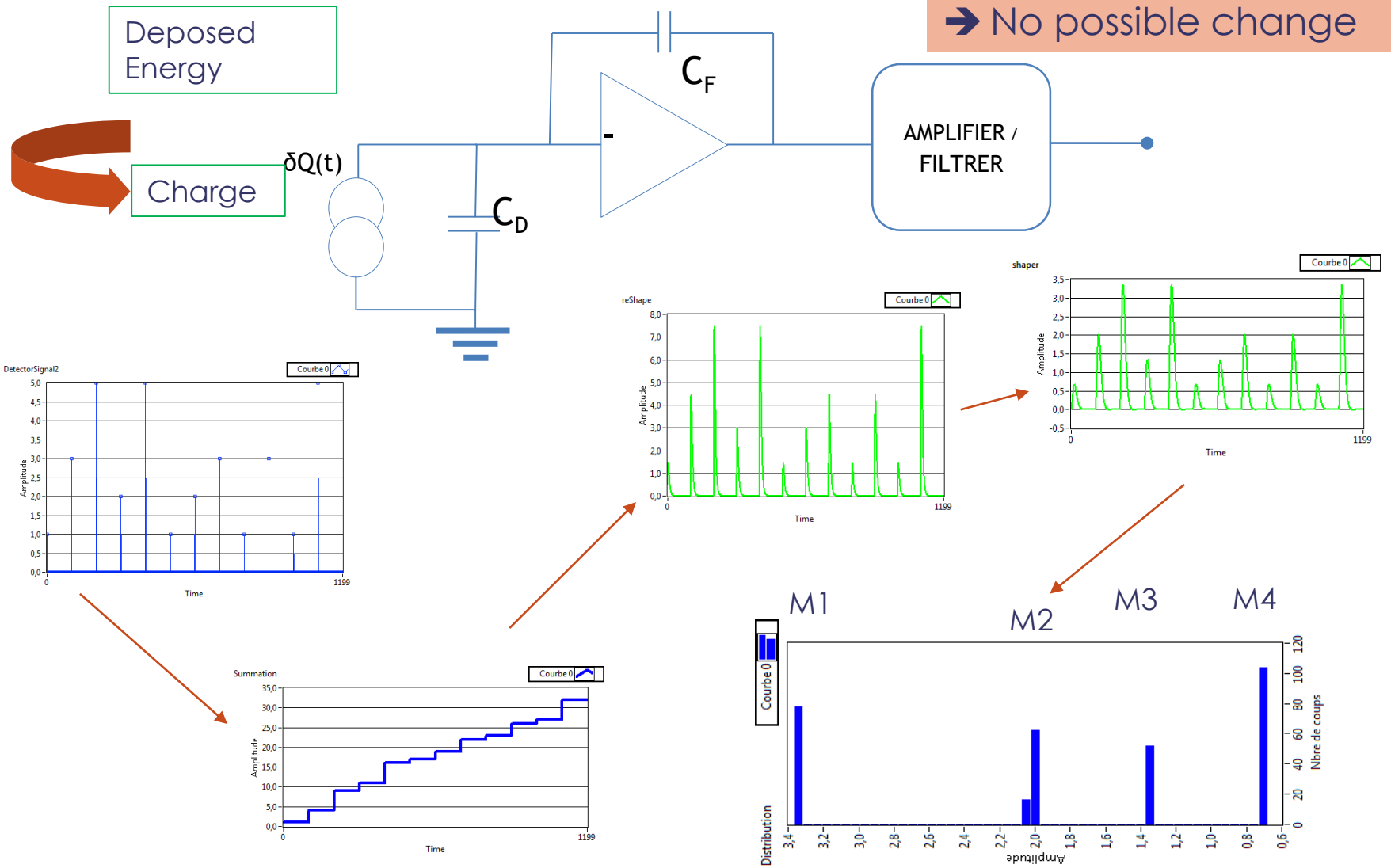
CSA, Current Preamplifier

Optimal filtering

➔ *To handle noise*

Signal Processing vs Available digital architecture

➔ Signal Processing fixe (few options)
➔ Few gain, few Shaping Time
➔ No possible change



Signal Processing vs Available digital architecture

Critères	CPU	CPU+AI-CoPro	GPU	FPGA	ASIC
Adaptability (to various model)	**High**	**High**	Moyen	**Low**	Aucune
Calculus power	**low**	Mean-High	**High**	**High**	Mean
Latency	Mean (ms)	Mean (ms)	**Faible** (µs)	**Faible**(10 ns)	**Très faible**
Input stream	**Low**	Mean	**High**	**High**	**High**
Parallelism	**Low**	Mean	**High**	**High**	**High**
Electrical power consumption	Mean	Mean	Mean	Mean	**High**
Easy to use	**Facile**	Moyen	Mean	**Complex**	**Very Complex**
Density of model	Moyen	**Low**	**Low**	**High**	**High**

— — — *Complexity to hold* + + +

RISC-V EcoSystem

- Initially a UC Berkeley research project in 2010, RISC-V is not a processor but an instruction set specification (ISA).
- RISC-V has been governed by the RISC-V Foundation since 2015 and then by RISC-V International in 2019.

RISC-V Ecosystem

Software

Open-source software:
Gcc, binutils, glibc, Linux, BSD, LLVM, QEMU, FreeRTOS, ZephyrOS, LiteOS, SylixOS, …

Commercial software:
Lauterbach, Segger, IAR, Micrium, ExpressLogic, Ashling, AntMicro, Imperas, UltraSoC …

RISC-V Foundation: ISA specification | Golden Model | Compliance

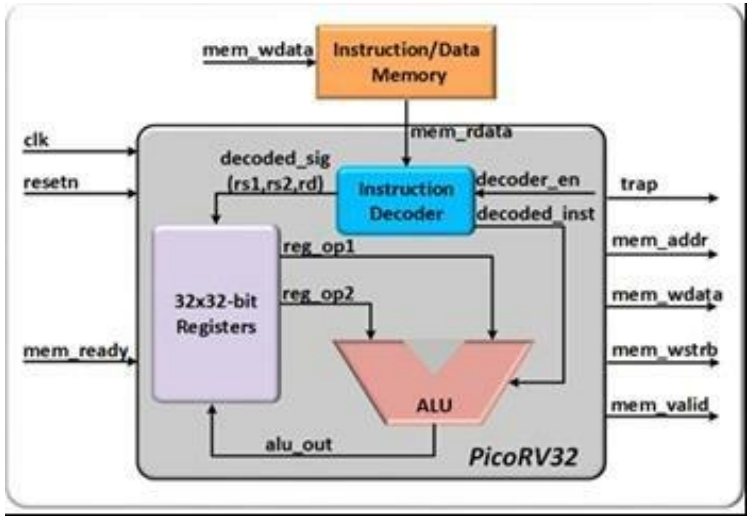
Hardware

Open-source cores:
Rocket, BOOM, RI5CY, Ariane, PicoRV32, Piccolo, SCR1, Shakti, Swerv, Hummingbird, …

Commercial core providers:
Andes, Bluespec, Cloudbear, Codasip, Cortus, C-Sky, InCore, Nuclei, SiFive, Syntacore, …

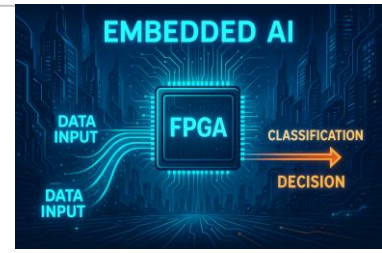
Inhouse cores:
Nvidia, +others

5



RV[SIZE] [EXTENSIONS] [Z_Extensions]

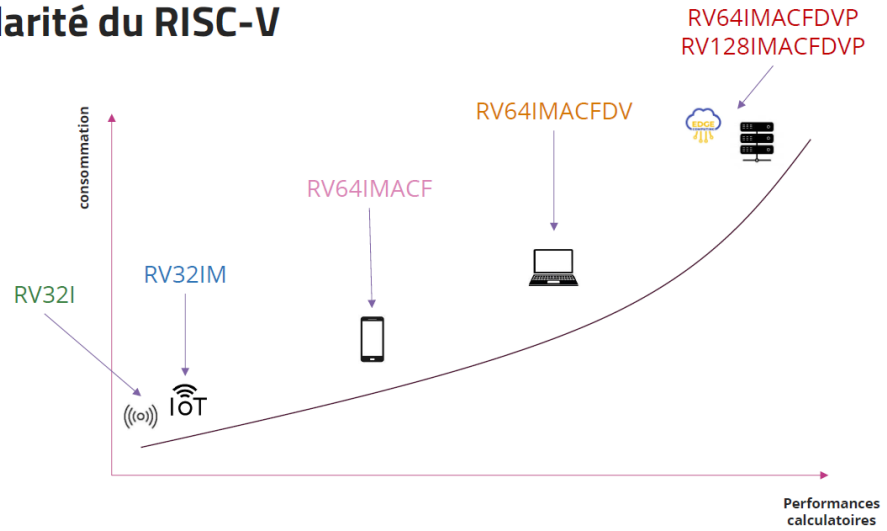
32b
64b
128b



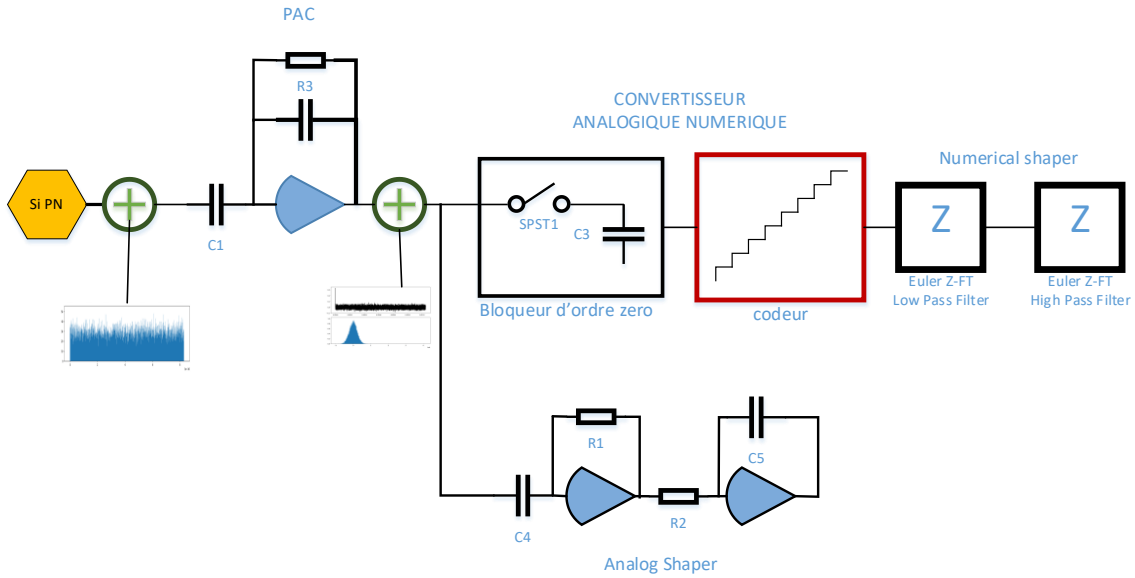
I: Basic extension present in all RISC-V processors
M: Integer multiplication and division (hardware accelerator)
A: Atomic memory operation (multiprocessor memory coherence)
C: Instruction compressed to 16 bits instead of 32 bits
F: Single-precision floating-point operations
D: Double-precision floating-point operations

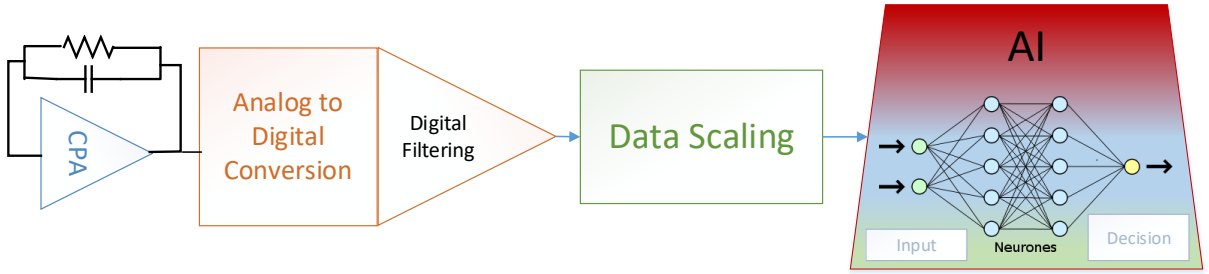
Zicsr: Non-standard extension: Status control register

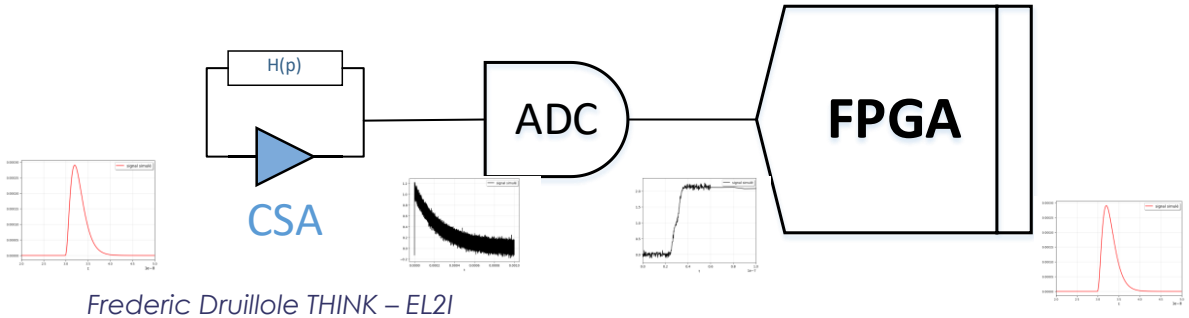
Modularité du RISC-V



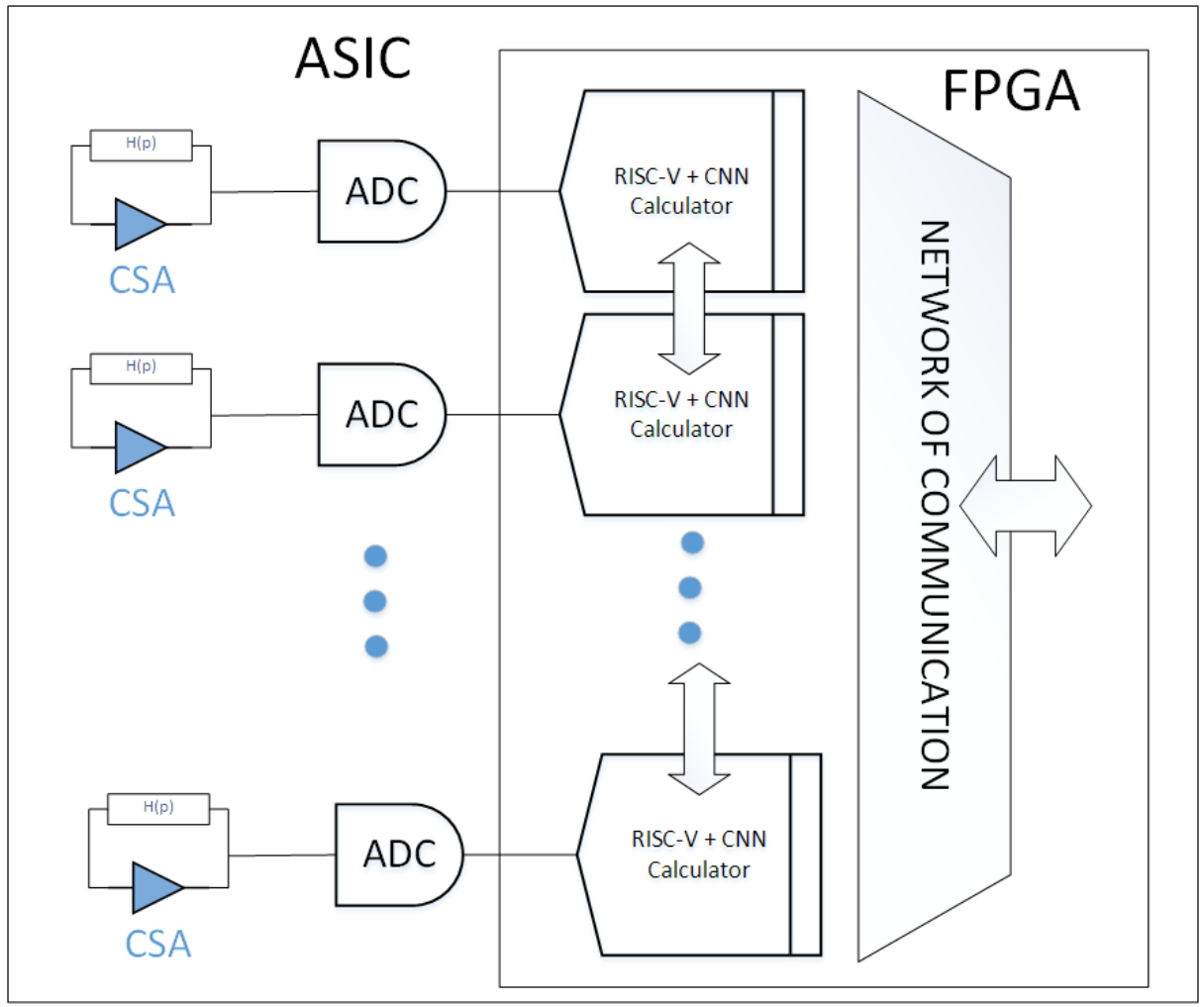
New Paradigme!







R&T OSCAR et ANR OSCARI



Solution

- **ADC: 200Mech/s x 12bits** ➔ 2,4Gbit/s ➔ 300MB/s
- 8 channels ➔ 19,2Gbit/s = 2,4GB/s

➔ **Besoin FIFO ou DMA DDR**

- Choice of External ADC: Parallel Interface (CMOS/LVDS) or JESD204B
- small RISC-V type RV32IMC or RV32IMAC (ex: picoRV32)

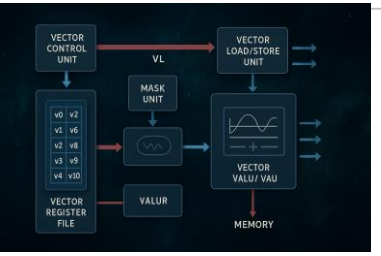
Notice: FIR/Conv1D/NN ➔ always same operations: MAC/DSP

$$z_i = \sum_{j=1}^{d} w_{ij} x_j + b_i, y_i = \sigma(z_i)$$

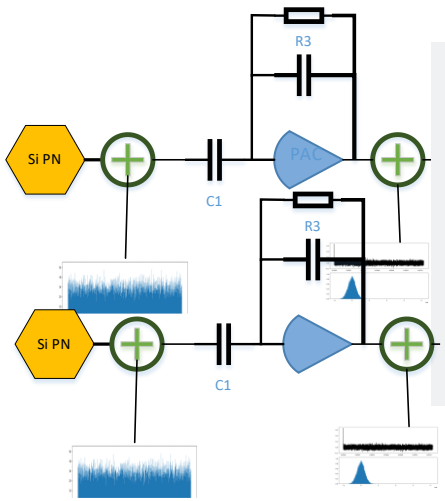
➔ **Matrix Multiplication**

➔ **RISC-V RV32IMC with Vector extension : It allows operations on data vectors to be performed in a single instruction.**

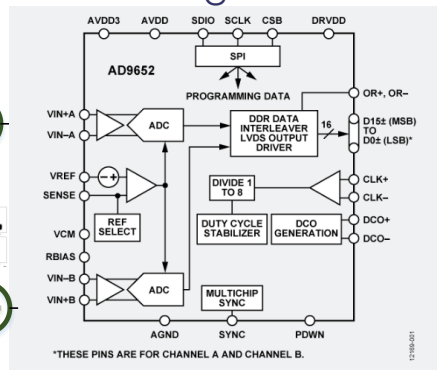
Solution FPGA OSCAR

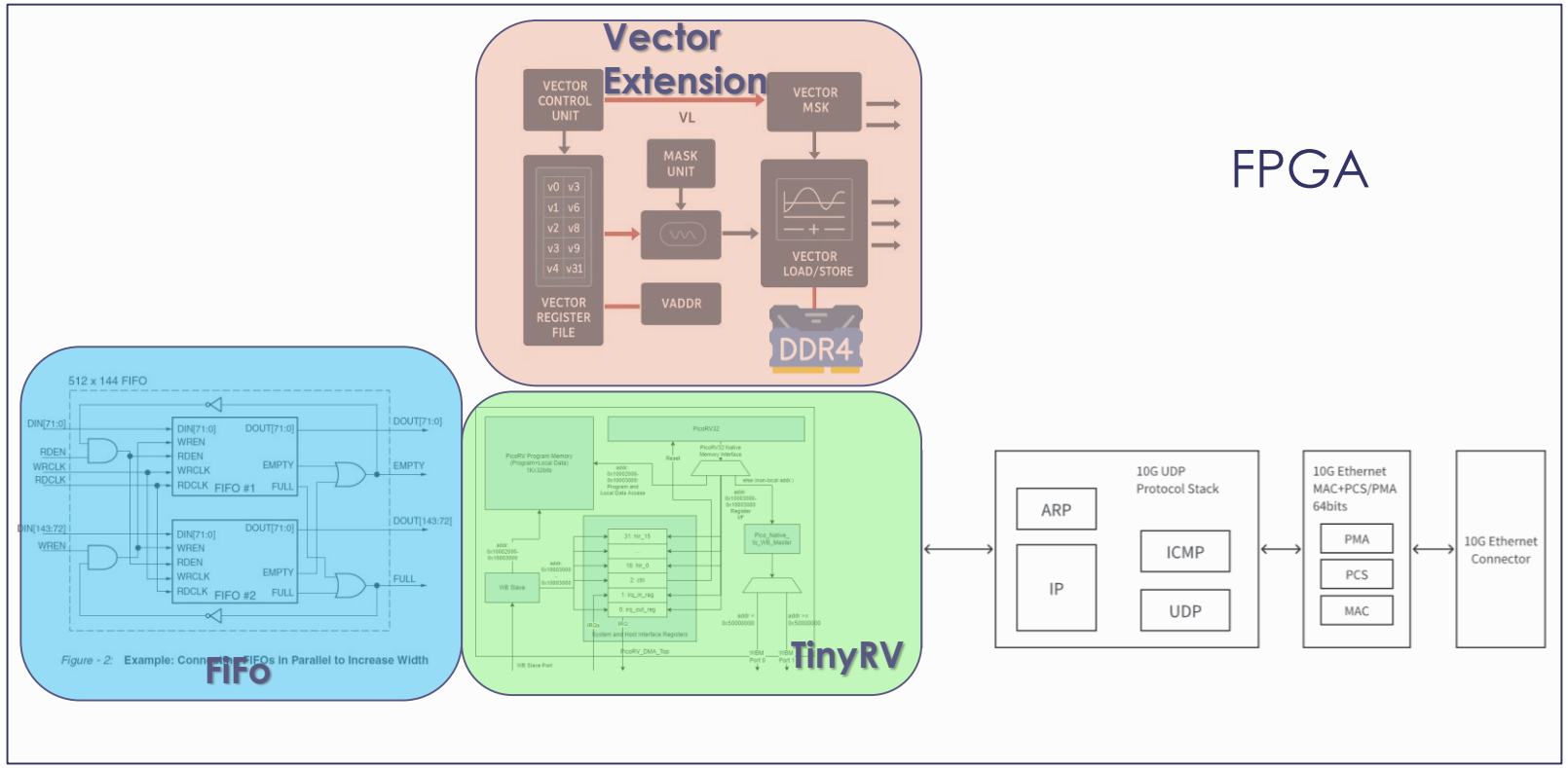


PAC



Analog Device



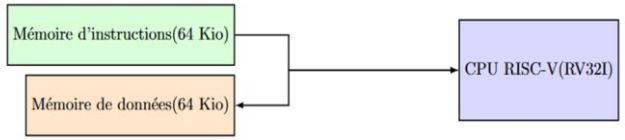


➔ R&T OSCAR ➔ Modélisation python
➔ Synthèse sur KINTEX
➔ Etude ADC µElec 65nm

OSCAR
Open System for capacitive Sensor aCcess & reAdout with aRtificial intelligence

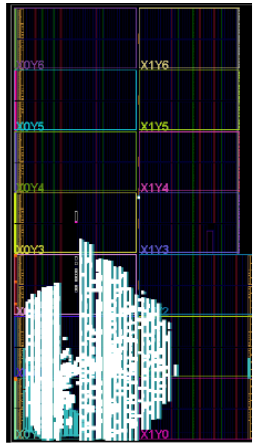
1 Digital Modeling	2 OSCAR-FACE Digital Twin	3 RISC-V Optimisation	4 SDK for Asymetric Multiprocessor Configuration	5 OSCARUS ASIC

Python SciPy Functional Modeling



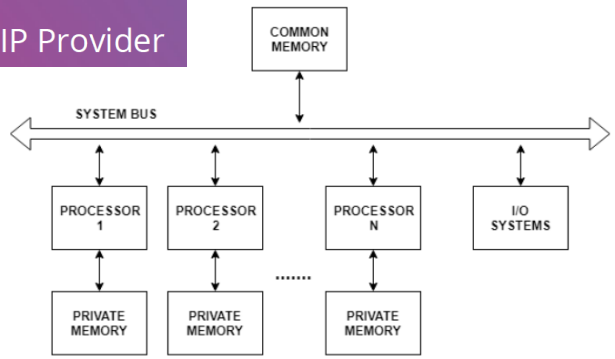
Organisation mémoire :
- Adressage sur 32 bits
- Organisation par octets
- Incrémentation par pas de 4 (32 bits)

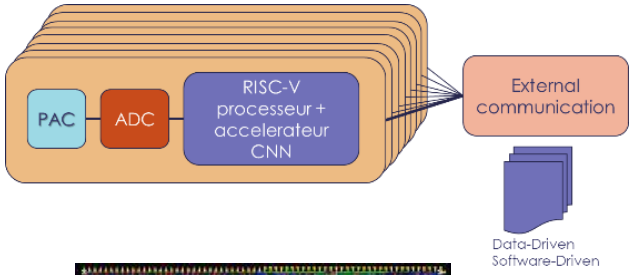
Figure 1: Architecture Harvard du processeur RISC-V (séparation mémoire instructions et données).

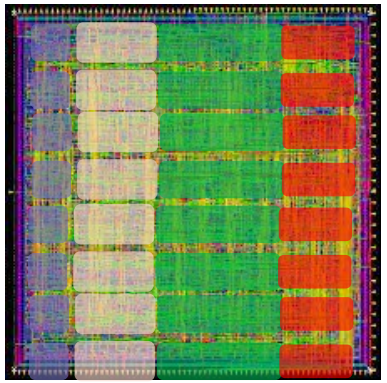


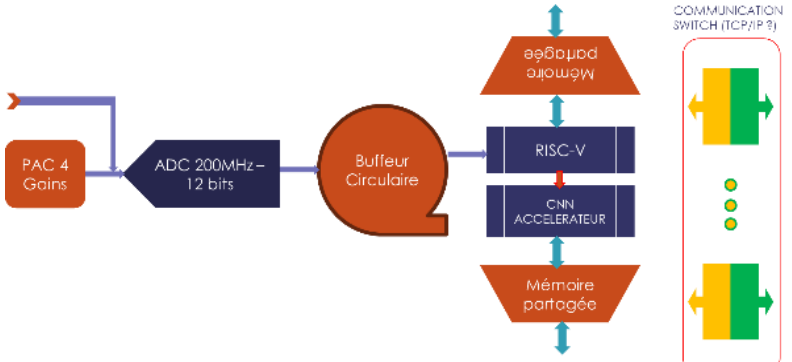
Keysom

RISC-V IP Provider









Keysom delivers a unique solution with a qualified team

Keysom is funded by four people willing to allow the semiconductor industry players to **benefit from custom core architectures quickly and easily,** to improve the PPA of their products.

Cyril Sagonero

CEO

Luca Testa

COO

Jérémie Crenne

HW CTO

Fabrice Bonnet

SW CTO

20 people, mostly PhD coming from R&D centers, specialized in computer science, processor architectures, compiler and formal verification

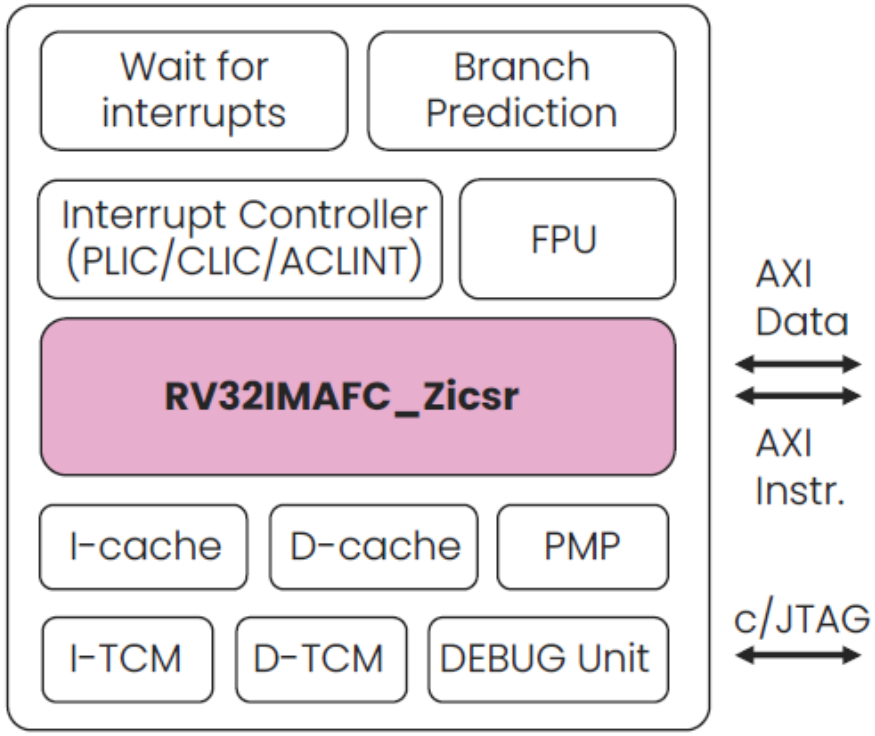
Tailored solutions for each customer, we want to understand your unique need

Keysom DNA: agile, better, faster

Based in **France**

https://keysom.io

Our RISC-V highly modular Cores Portfolio – millions of configurations

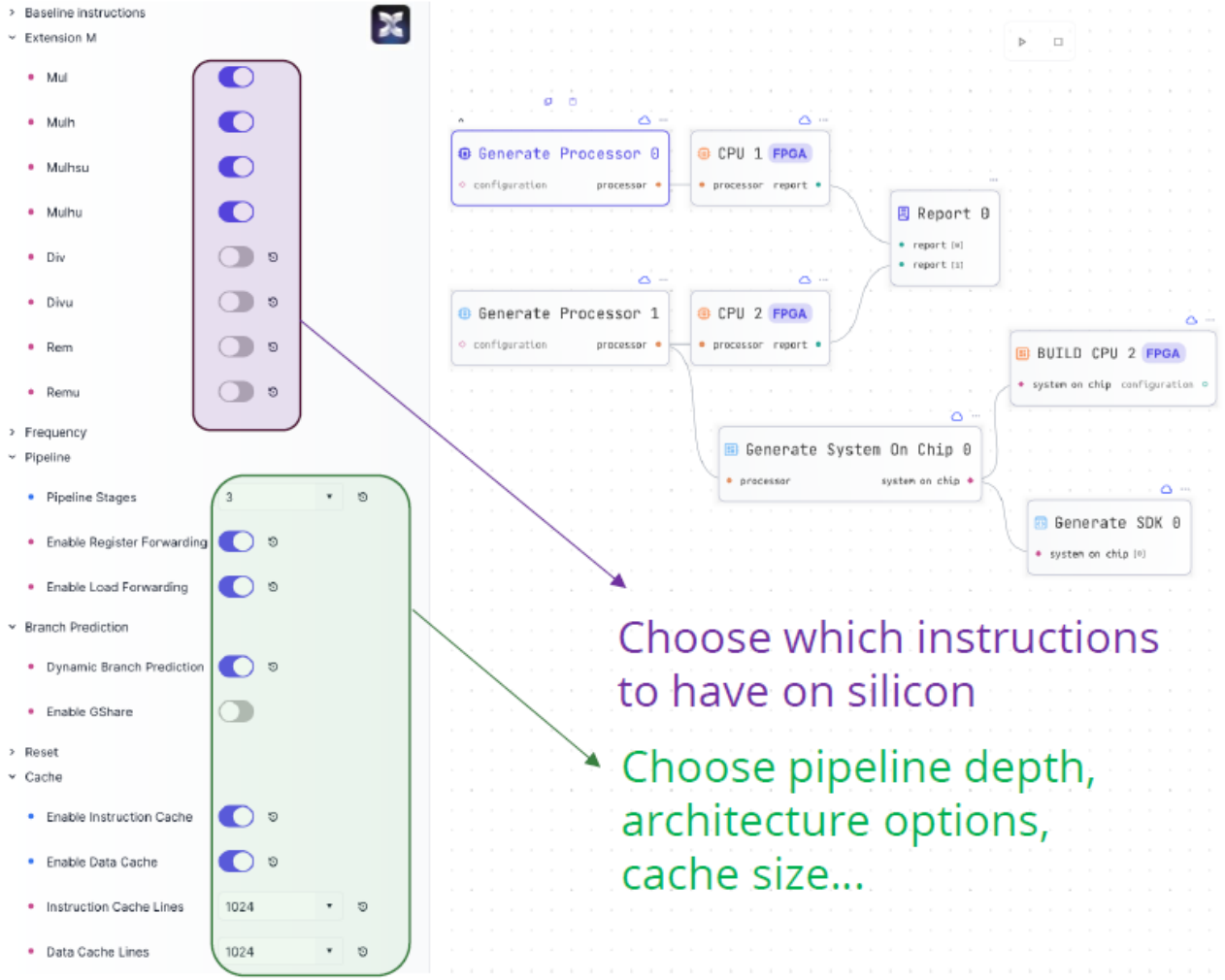


- *Configurable* pipeline stages, from 1 to 5
- Any block, feature and instruction is optional
- Smart LLVM Compiler
- *Configurable* Hardware breakpoints configurable
- *Configurable* Watchpoints configurable
- Privileged levels (User, Supervisor, Machine modes)

✓ Supported operating systems: freeRTOS Zephyr ECLIPSE THREADX

https://keysom.io

CoreXplorer® – Our Design Exploration Tool



- Adaptable according to your priorities (Power, Performance, Area)
- A CPU architecture generated **in minutes** without any hardware expertise
- A turnkey solution : **SW development environment** delivered with the HW
- Cloud or On-Premise

https://keysom.io

Focus on our specific features

Optimized LLVM

- Supports the most recent stable LLVM version
- Improvement of the LLVM backend for Keysom cores, in order to improve LLVM results when less efficient than GCC
- Automatic emulation of any instruction drastically lowering the silicon area needed for core implementation

Formal Verification

- Unlock verification with theorem proving when model-checking cannot succeed
- Thanks to its expertise in ROCQ, Keysom formally proves equivalence between its own implementation and the standardized formal model of the architecture from SAIL

Edge AI deployment

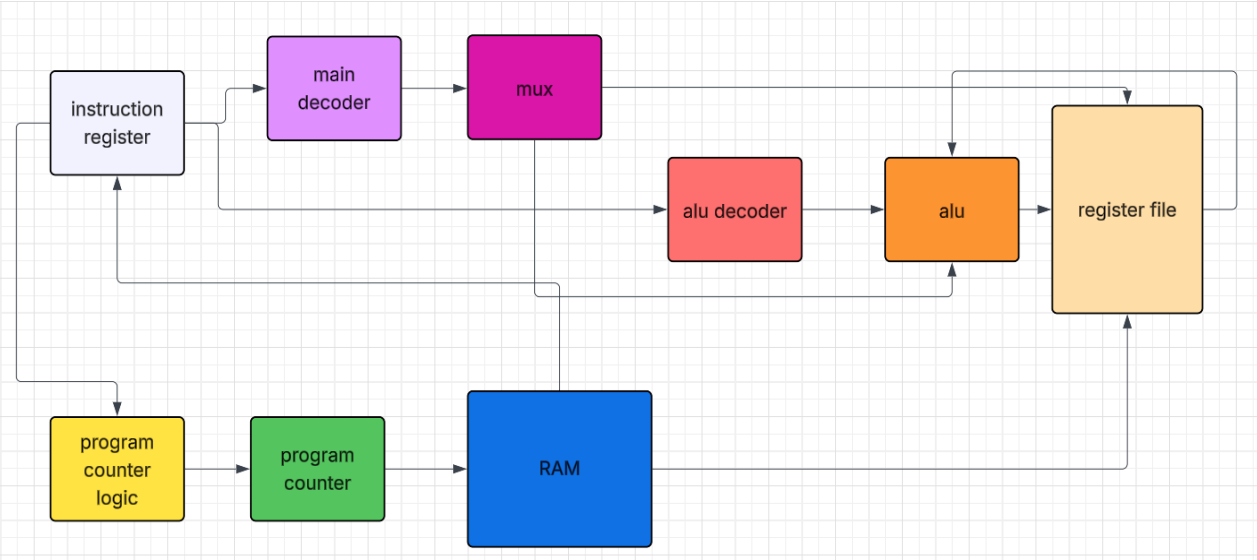
- Choose a small AI model from a library
- Keysom framework automatically adapts the model to the chosen core
- The AI model is deployed on the core without effort

https://keysom.io

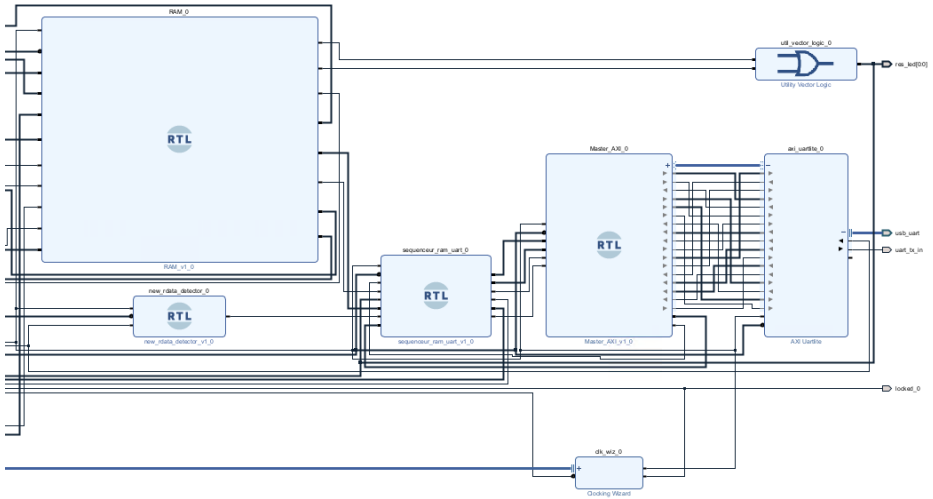
NAIVE TRY

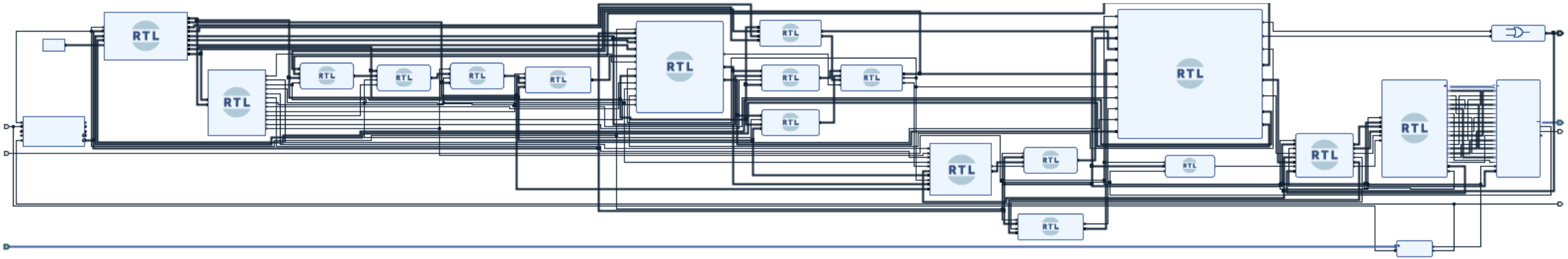
Internship 2025

Harvard architecture, which means that instruction memory and data memory are separate.



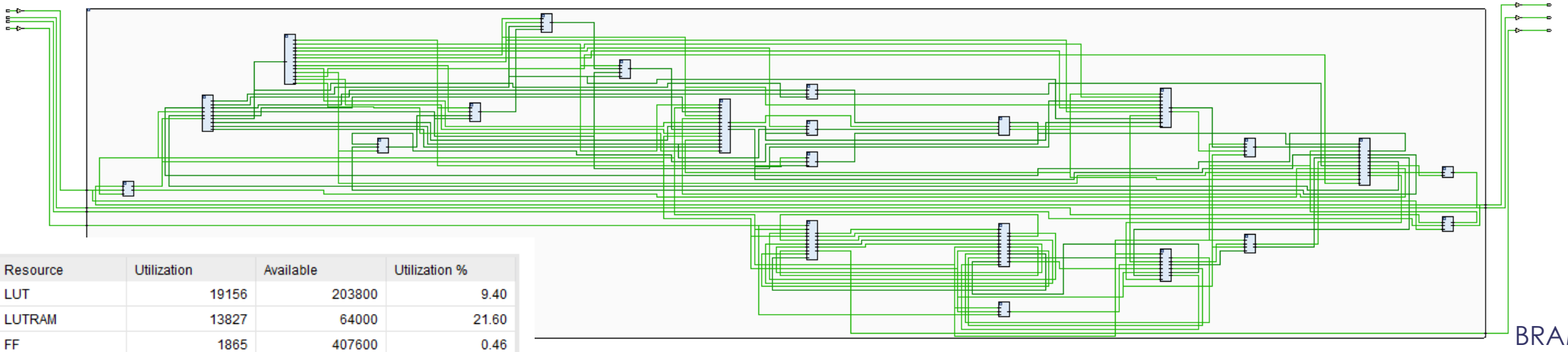
Communication with UARTLITE



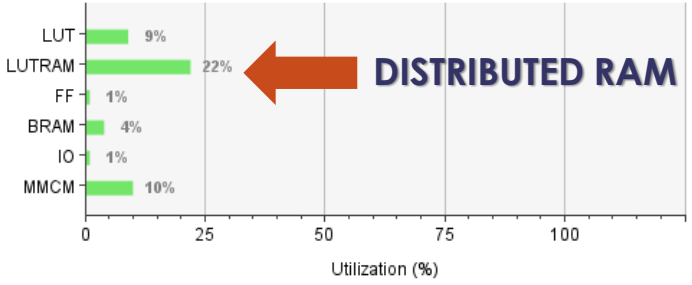


NAIVE TRY

Internship 2025



Resource	Utilization	Available	Utilization %
LUT	19156	203800	9.40
LUTRAM	13827	64000	21.60
FF	1865	407600	0.46
BRAM	16	445	3.60
IO	7	500	1.40
MMCM	1	10	10.00



DISTRIBUTED RAM

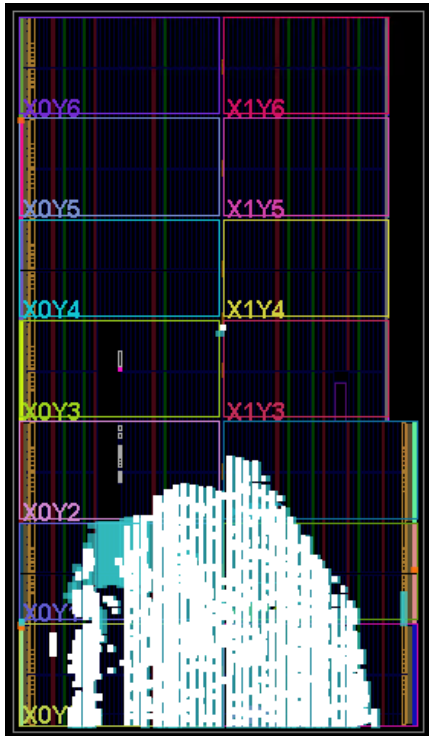
Name	Slice LUTs (203800)	Slice Registers (407600)	F7 Muxes (101900)	F8 Muxes (50950)	Slice (50950)	LUT as Logic (203800)	LUT as Memory (64000)	Block RAM Tile (445)	Bonded IOB (500)	IBUFDS (480)	BUFGCTRL (32)	MMCME2_ADV (10)
N SimpleCycle_RISC_wrapper	19156	1865	1474	514	5538	5329	13827	16	7	1	4	1
SimpleCycle_RISC_i (SimpleCycle_RISC)	19156	1865	1474	514	5538	5329	13827	16	0	1	4	1
ALU_0 (SimpleCycle_RISC_ALU_0_0)	552	0	0	0	159	552	0	0	0	0	0	0
alu_decoder_0 (SimpleCycle_RISC_alu_	34	0	0	0	14	34	0	0	0	0	0	0
alu_operand_a_mux_0 (SimpleCycle_RIS	16	0	0	0	12	16	0	0	0	0	0	0
auipc_mux_adder_0 (SimpleCycle_RISC_	48	0	0	0	19	48	0	0	0	0	0	0
axi_uartlite_0 (SimpleCycle_RISC_axi_ua	83	100	0	0	36	73	10	0	0	0	0	0
clk_wiz_0 (SimpleCycle_RISC_clk_wiz_0_	0	0	0	0	0	0	0	0	0	1	2	1
instruction_register_0 (SimpleCycle_RISC	38	64	0	0	40	38	0	0	0	0	0	0
instruction_uart_reg_0 (SimpleCycle_RIS	42	62	0	0	43	42	0	0	0	0	0	0
jump_multiplexor_0 (SimpleCycle_RISC_	33	0	0	0	26	33	0	0	0	0	0	0
main_decoder_0 (SimpleCycle_RISC_ma	15	0	0	0	13	15	0	0	0	0	0	0
Master_AXI_0 (SimpleCycle_RISC_Maste	11	14	0	0	8	11	0	0	0	0	0	0
multiplexeur_source_0 (SimpleCycle_RIS	16	0	0	0	10	16	0	0	0	0	0	0
mux_imm_rs_1 (SimpleCycle_RISC_mux	32	0	0	0	17	32	0	0	0	0	0	0
new_rdata_detector_0 (SimpleCycle_RIS	53	38	0	0	14	53	0	0	0	0	0	0
proc_sys_reset_0 (SimpleCycle_RISC_pr	14	25	0	0	9	13	1	0	0	0	0	0
program_counter_0 (SimpleCycle_RISC_	1	32	0	0	11	1	0	0	0	0	0	0
program_counter_logic_0 (SimpleCycle_	91	33	0	0	39	91	0	0	0	0	1	0
RAM_0 (SimpleCycle_RISC_RAM_0_0)	17364	288	1218	514	4717	3548	13816	16	0	0	1	0
regfile_write_data_s_0 (SimpleCycle_RIS	16	0	0	0	14	16	0	0	0	0	0	0
register_file_0 (SimpleCycle_RISC_regis	621	1089	256	0	490	621	0	0	0	0	0	0
		119	0	0	56	82	0	0	0	0	0	0
		0	0	0	1	1	0	0	0	0	0	0

ALU

RAM

Name	Waveform	Period (ns)	Frequency (MHz)
sys_diff_clock_clk_p	{0.000 2.500}	5.000	200.000
clk_out1_SimpleCycle_RISC_clk_wiz_0_1	{0.000 10.000}	20.000	50.000
clkfbout_SimpleCycle_RISC_clk_wiz_0_1	{0.000 2.500}	5.000	200.000

BRAM + LUTBRAM



Conclusion

- To manage the challenge, we have to master the quantity of RAM.
- To speed calculation, we need to compress instructions
- To speed calculation, we need to use Vector Extension and optimize the model (THINK)

→ Keysom Product → **KINTEX (LICENSE)**

→ **ASIC IP**

- Build a High-speed ADC from start (65nm).
- Build RISC-V multi parallel processor system with OS.
- Explore a rupture Processing architecture to optimized instruments.