Computational SRAM: Towards Efficient Near-Memory Computing through Tightly Coupled HW/SW Design

Emanuele Valea, *Jean-Philippe Noel, Thaddée Bricout, Henri-Pierre Charles, Leo De La Fuente, Bastien Giraud, Maha Kooli, Benjamin Lacour, Manuel Pezzin, Maria Ramirez Corrales*

2nd In-Memory Architectures and Computing Applications Workshop

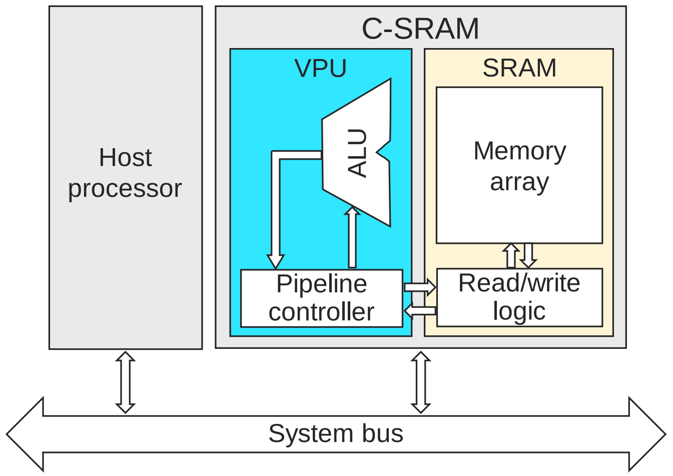
Computational SRAM (C-SRAM)

- The proposed **NMC** solution is compatible with any existing memory macros (**off-the-shelf memory compilers, custom design**)
- Coupled with a **Vector Processing Unit (VPU)** executing NMC instructions issued from the host CPU
- A **specific C-SRAM ISA** is supported
- In SoC architectures **data RAM** can be advantageously **replaced by a C-SRAM**

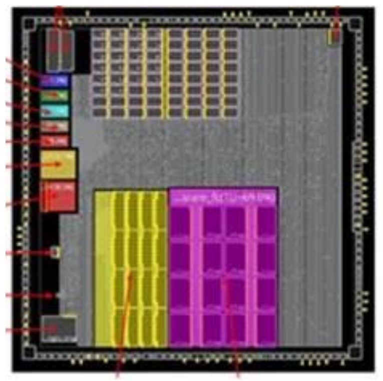
C-SRAM Instruction Set Architecture

Category	Mnemonic	Description
Memory	copy	Copy a line into another
	bcast	Broadcast 8/16/32-bit value to the whole Line
	hswap	Horizontal 32/64-bit word swap
Logical	slli, srli	Shift Left or Right Logical Immediate
	(n)and, (n)or, (n)xor	Logical AND, OR & XOR (and negation)
Arithmetic	add, sub	Arithmetic 8/16/32-bit Addition & Subtraction
	mullo, mulhi	Arithmetic 8-bit integer Multiply
	maclo	Arithmetic 8-bit integer Multiply-Accumulate

C-SRAM Architecture



C-SRAM Testchip

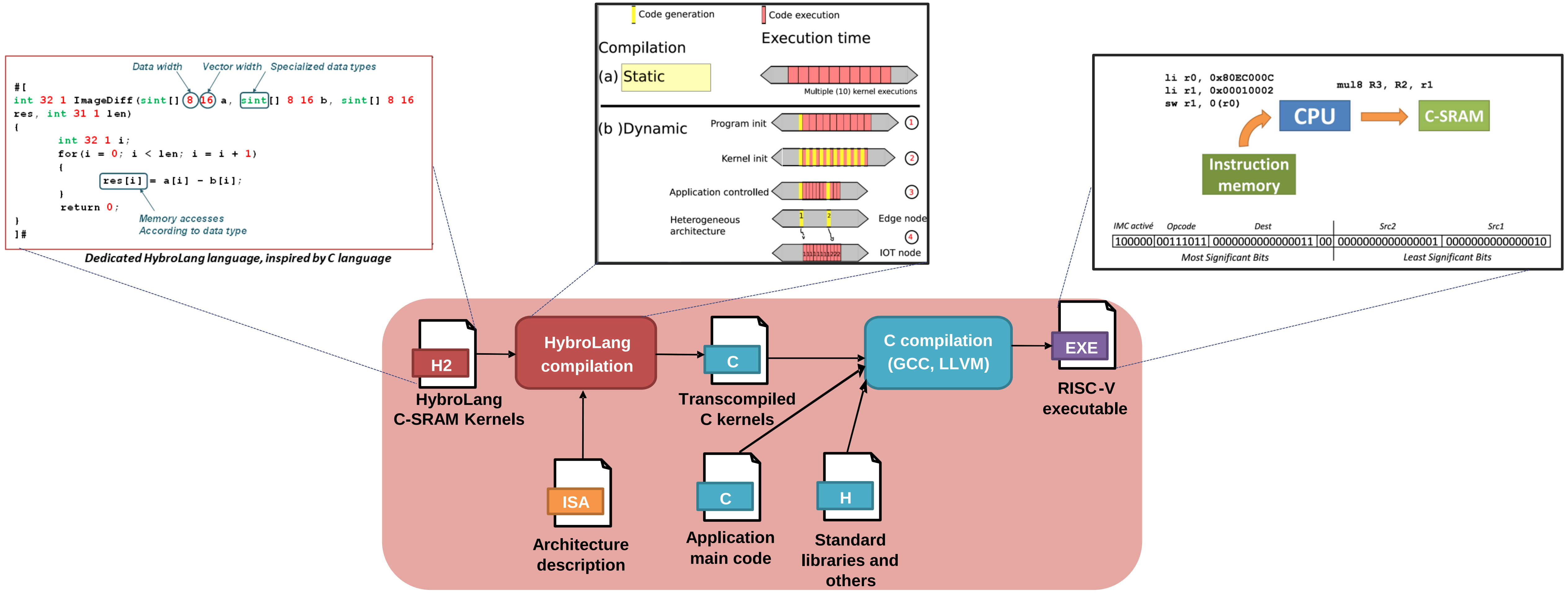


- What about programming and **SW compatibility**?

[1] M. Kooli *et al.*, "*Towards a Truly Integrated Vector Processing Unit for Memory-bound Applications Based on a Cost-competitive Computational SRAM Design Solution",* ACM JETC, Vol. 18, Issue 2, No. 40, 2022, pp. 1-26.
[2] J.-P. Noel *et al.*, "*A 35.6 TOPS/W/mm2 3-stage pipelined computational SRAM with adjustable form factor for highly data-centric applications",* IEEE SSCL, Vol. 3, 2020, pp. 286–289.
[3] J.-P. Noel *et al.*, "*Computational SRAM Design Automation using Pushed-Rule Bitcells for Energy-Efficient Vector Processing"*, DATE Conference, 2020, pp. 1187–1192.

HybroGen compilation toolchain (open-source)

- **HybroGen** provides a **compilation flow** from source C code to **parallelized** binary employing specific **NMC instructions**
- **Dynamic compilation** approach in order to **adapt NMC scheduling** to format and size of input data



```
#[
int 32 1 ImageDiff(sint[] 8 16 a, sint[] 8 16 b, sint[] 8 16
res, int 31 1 len)
{
        int 32 1 i;
        for(i = 0; i < len; i = i + 1)
        {
                res[i] = a[i] - b[i];
        }
        return 0;
}
]#
```

Dedicated HybroLang language, inspired by C language

```
li r0, 0x80EC000C
li r1, 0x00010002
sw r1, 0(r0)
```

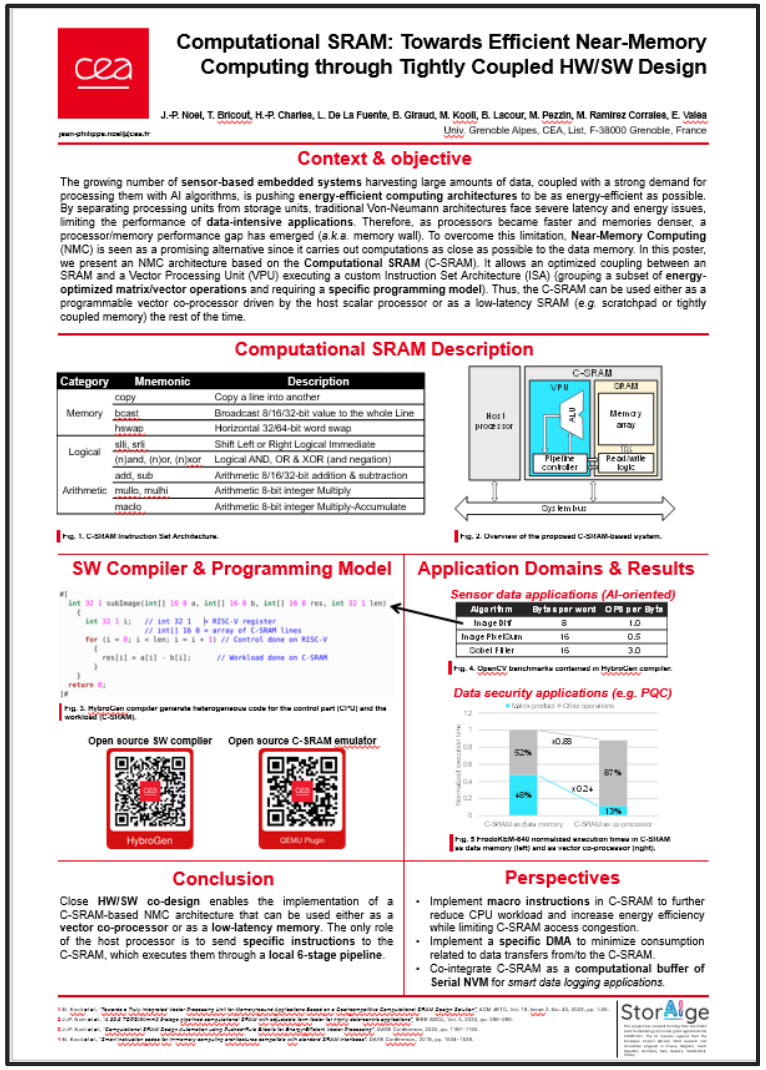
```
mul8 R3, R2, r1
```

IMC activé	Opcode	Dest		Src2	Src1
100000	00111011	0000000000000011	00	0000000000000001	0000000000000010

Most Significant Bits — *Least Significant Bits*

[1] M. Kooli *et al.*, "*Towards a Truly Integrated Vector Processing Unit for Memory-bound Applications Based on a Cost-competitive Computational SRAM Design Solution*", ACM JETC, Vol. 18, Issue 2, No. 40, 2022, pp. 1-26.
[4] K. Mambu et al., "*Dedicated Instruction Set for Pattern-based Data Transfers: an Experimental Validation on Systems Containing In-Memory Computing Units*," in IEEE TCAD

Come to the poster!



Computational SRAM: Towards Efficient Near-Memory Computing through Tightly Coupled HW/SW Design

J.-P. Noel, T. Bricout, H.-P. Charles, L. De La Fuente, B. Giraud, M. Kooli, B. Lacour, M. Pezzin, M. Ramirez Corrales, E. Valea
Univ. Grenoble Alpes, CEA, List, F-38000 Grenoble, France
jean-philippe.noel@cea.fr

Context & objective

The growing number of **sensor-based embedded systems** harvesting large amounts of data, coupled with a strong demand for processing them with AI algorithms, is pushing **energy-efficient computing architectures** to be as energy-efficient as possible. By separating processing units from storage units, traditional Von-Neumann architectures face severe latency and energy issues, limiting the performance of **data-intensive applications**. Therefore, as processors became faster and memories denser, a processor/memory performance gap has emerged (*a.k.a.* memory wall). To overcome this limitation, **Near-Memory Computing** (NMC) is seen as a promising alternative since it carries out computations as close as possible to the data memory. In this poster, we present an NMC architecture based on the **Computational SRAM** (C-SRAM). It allows an optimized coupling between an SRAM and a Vector Processing Unit (VPU) executing a custom Instruction Set Architecture (ISA) (grouping a subset of **energy-optimized matrix/vector operations** and requiring a **specific programming model**). Thus, the C-SRAM can be used either as a programmable vector co-processor driven by the host scalar processor or as a low-latency SRAM (*e.g.* scratchpad or tightly coupled memory) the rest of the time.

Computational SRAM Description

Category	Mnemonic	Description
Memory	copy	Copy a line into another
	bcast	Broadcast 8/16/32-bit value to the whole Line
	hswap	Horizontal 32/64-bit word swap
Logical	slli, srli	Shift Left or Right Logical Immediate
	(n)and, (n)or, (n)xor	Logical AND, OR & XOR (and negation)
Arithmetic	add, sub	Arithmetic 8/16/32-bit addition & subtraction
	mullo, mulhi	Arithmetic 8-bit integer Multiply
	maclo	Arithmetic 8-bit integer Multiply-Accumulate

Fig. 1. C-SRAM Instruction Set Architecture.

Host processor | C-SRAM | VPU | ALU | Pipeline controller | SRAM | Memory array | Read/write logic | System bus

Fig. 2. Overview of the proposed C-SRAM-based system.

SW Compiler & Programming Model

```
#[
  int 32 1 subImage(int[] 16 8 a, int[] 16 8 b, int[] 16 8 res, int 32 1 len)
    {
      int 32 1 i;    // int 32 1   = RISC-V register
                     // int[] 16 8 = array of C-SRAM lines
      for (i = 0; i < len; i = i + 1) // Control done on RISC-V
        {
          res[i] = a[i] - b[i];       // Workload done on C-SRAM
        }
    }
  return 0;
]#
```

Fig. 3. HybroGen compiler generate heterogeneous code for the control part (CPU) and the workload (C-SRAM).

Open source SW compiler — HybroGen

Open source C-SRAM emulator — QEMU Plugin

Application Domains & Results

Sensor data applications (AI-oriented)

Algorithm	Bytes per word	OPS per Byte
Image Diff	8	1.0
Image PixelSum	16	0.5
Sobel Filter	16	3.0

Fig. 4. OpenCV benchmarks contained in HybroGen compiler.

Data security applications (e.g. PQC)

Matrix product | Other operations | Normalized execution time | 52% | 48% | 87% | 13% | x0.88 | x0.24 | C-SRAM as data memory | C-SRAM as co-processor

Fig. 5 FrodoKEM-640 normalized execution times in C-SRAM as data memory (left) and as vector co-processor (right).

Conclusion

Close **HW/SW co-design** enables the implementation of a C-SRAM-based NMC architecture that can be used either as a **vector co-processor** or as a **low-latency memory**. The only role of the host processor is to send **specific instructions** to the C-SRAM, which executes them through a **local 6-stage pipeline**.

Perspectives

- Implement **macro instructions** in C-SRAM to further reduce CPU workload and increase energy efficiency while limiting C-SRAM access congestion.
- Implement a **specific DMA** to minimize consumption related to data transfers from/to the C-SRAM.
- Co-integrate C-SRAM as a **computational buffer of Serial NVM** for *smart data logging applications*.

[1] M. Kooli et al., "Towards a Truly Integrated Vector Processing Unit for Memory-bound Applications Based on a Cost-competitive Computational SRAM Design Solution", ACM JETC, Vol. 18, Issue 2, No. 40, 2022, pp. 1-26.
[2] J.-P. Noel et al., "A 35.6 TOPS/W/mm² 3-stage pipelined computational SRAM with adjustable form factor for highly data-centric applications", IEEE SSCL, Vol. 3, 2020, pp. 286-289.
[3] J.-P. Noel et al., "Computational SRAM Design Automation using Pushed-Rule Bitcells for Energy-Efficient Vector Processing", DATE Conference, 2020, pp. 1187-1192.
[4] M. Kooli et al., "Smart instruction codes for in-memory computing architectures compatible with standard SRAM interfaces", DATE Conference, 2018, pp. 1634-1639.

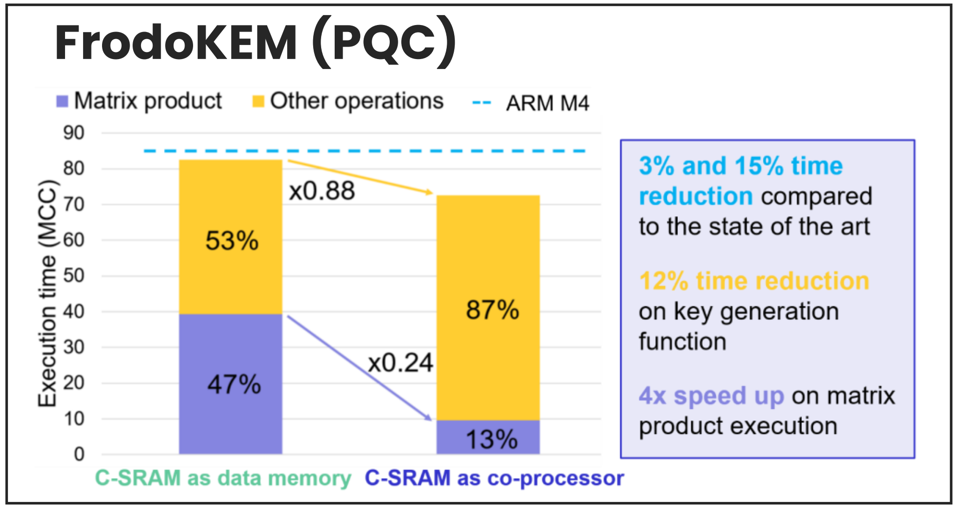
StorAIge

[illegible]

- We show some **preliminary results** on several applications:
  - Image processing
  - Artificial Intelligence
  - Cryptography (post-quantum)

	Sobel Filter (Edge Detection)	Matrix Multiplication	Frame Differencing	Convolution (Tensor Flow)
Speed-Up	~x4.3	~x4.9	~x7.7	~x3.5
Energy Reduction	~x8.3	~x13	~x10.3	~x4

[5] Mambu et al. 2022 *Towards Integration of a Dedicated Memory Controller and Its Instruction Set to Improve Performance of Systems Containing Computational SRAM.J. Low Power Electron. Appl.*,*12*, 18.



Thank you!

Emanuele Valea
Email: emanuele.valea@cea.fr