



Smart Instruction Codes For In-Memory Computing Architectures Compatible With Standard Sram Interfaces

Henri-Pierre Charles, Maha Kooli, Clément Touzet, Bastien Giraud,
Jean-Philippe Noel

► To cite this version:

Henri-Pierre Charles, Maha Kooli, Clément Touzet, Bastien Giraud, Jean-Philippe Noel. Smart Instruction Codes For In-Memory Computing Architectures Compatible With Standard Sram Interfaces. 2018. cea-01757665

HAL Id: cea-01757665

<https://cea.hal.science/cea-01757665>

Submitted on 3 Apr 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



SMART INSTRUCTION CODES FOR IN-MEMORY COMPUTING ARCHITECTURES COMPATIBLE WITH STANDARD SRAM INTERFACES

Maha Kooli, Henri-Pierre Charles, Clément Touzet, Bastien Giraud, Jean-Philippe Noel

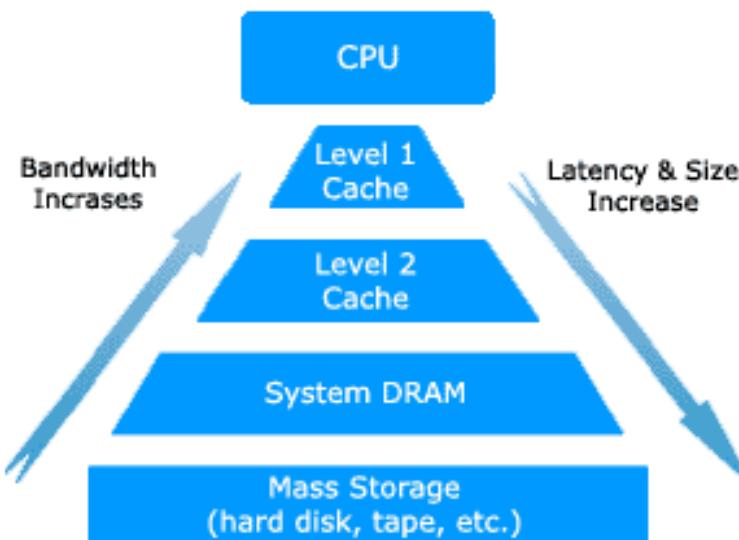
Univ. Grenoble Alpes, CEA, LETI/LIST, FRANCE

DATE'18

Dresden, Germany, March 22nd, 2018

BREAK THE MEMORY WALL! BUT HOW...?

"memory wall" or "funnel effect"...

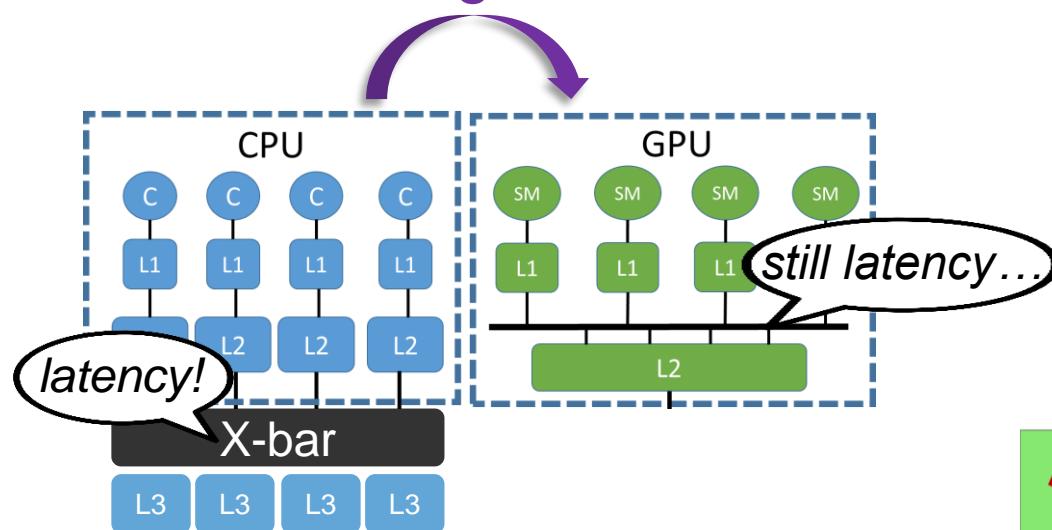


...is nowaday the main limitation for **high performance computing**

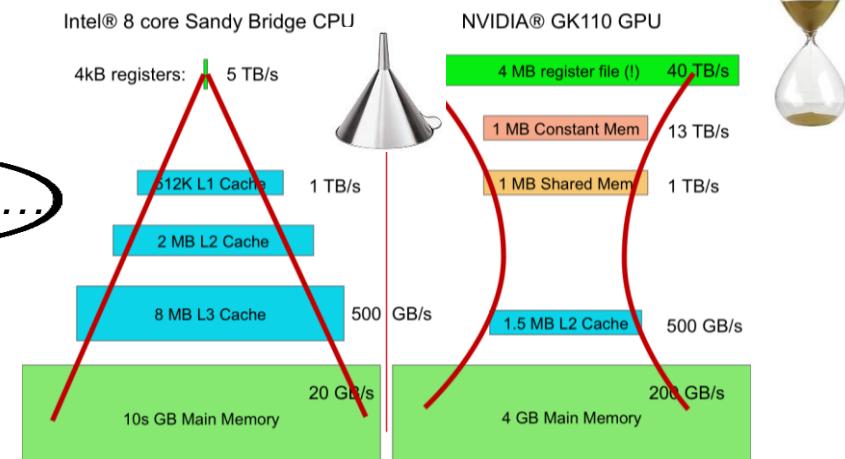
BOTTLENECK LIES IN THE MEMORY HIERARCHY

Let's do multi-core processors!

seems a good idea!



Source: Barcelona Supercomputing Center



Source: nVidia

Memory access is STILL a bottleneck, even in GPUs...

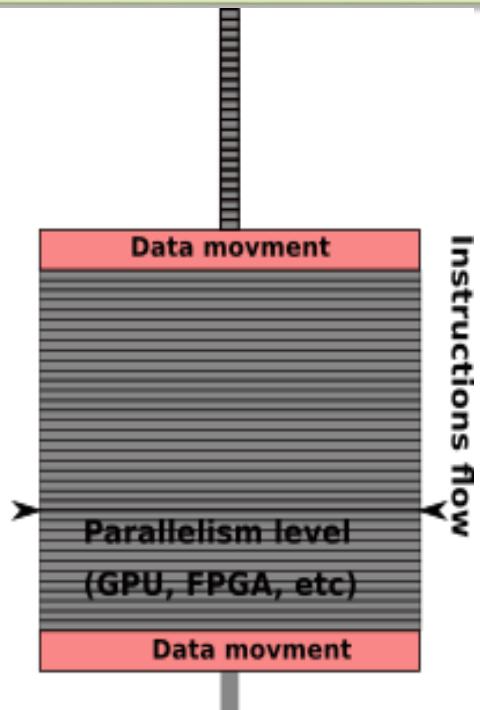
AMDALL LAW IS FOR EVERYBODY

GPU computing model (SPMD)

need to:

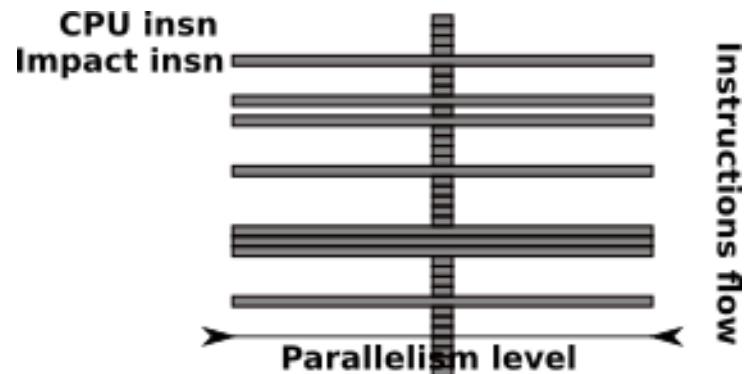
- Copy/transfer data
- Group parallel instructions

Sequential part



IMPACT computing model :

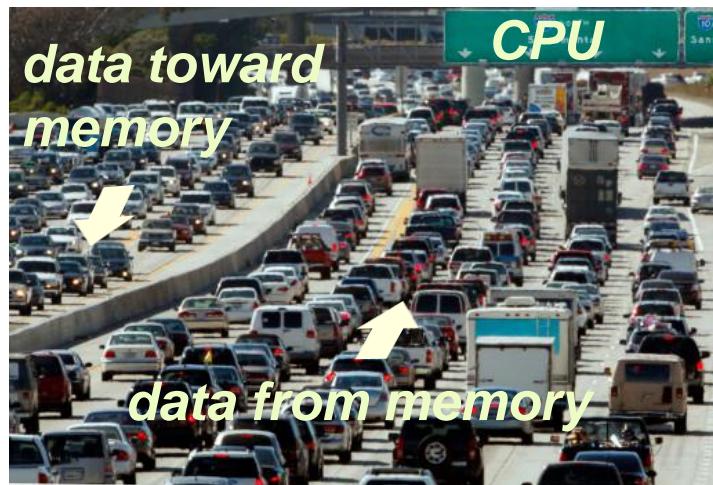
- No copy / transfer
- Fine grain parallel scalar interleaving



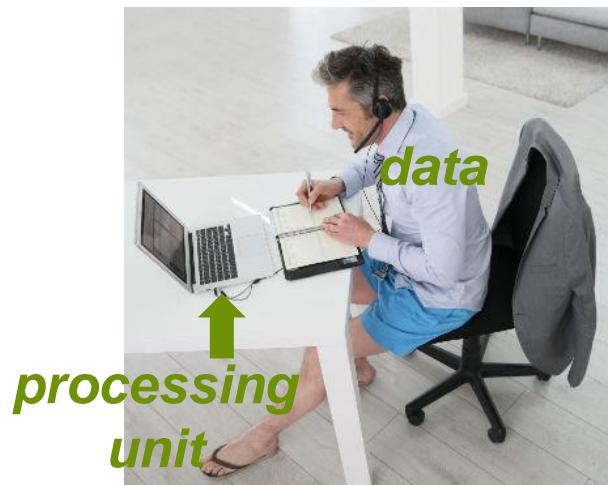
BRING THE COMPUTATION INTO MEMORY

When data start to look like motorists in the *traffic jam* during the rush hour...
(data @memory ↔ data @comp_unit)

Von Neumann world...



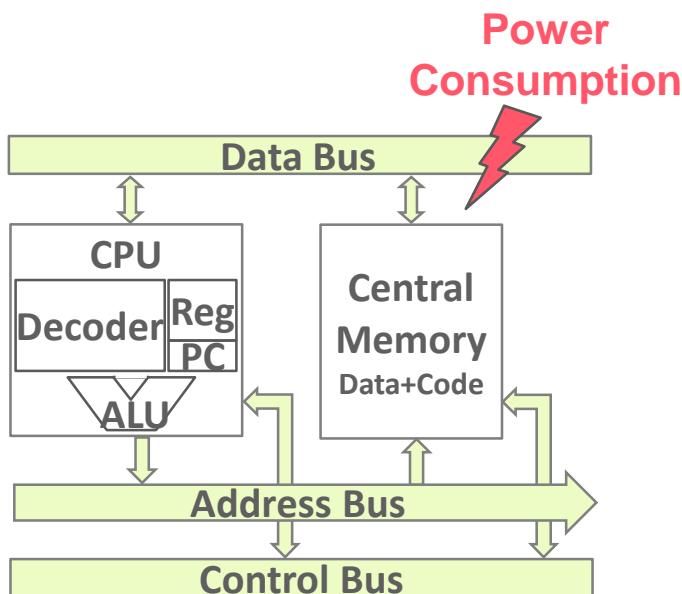
...meanwhile in IMC world!



...it's time to consider *teleworking*, in other word the **in-memory computing**

Von Neumann Model:

- Data & instruction in the same memory
- i.e. instructions are data
- SoC or PCB



Memory INSN: $ld\ r1 = @r2$

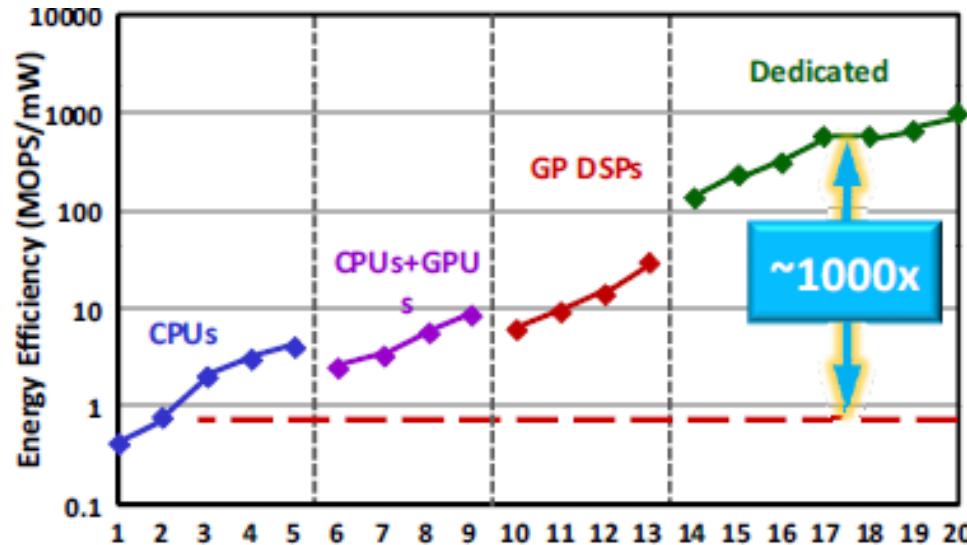
- 1 memory access (for the instruction)
- 1 instruction cycle (Decode + RF + memory access)

Compute INSN: $add\ r1 = r2 + r3$

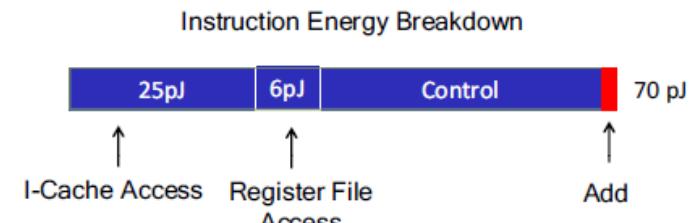
- Compute instruction
- 1 memory access (for the instruction)
- 1 computation (Decode + RF + ALU)

POWER WALL

M. Horowitz (Stanford), ISSCC 2014



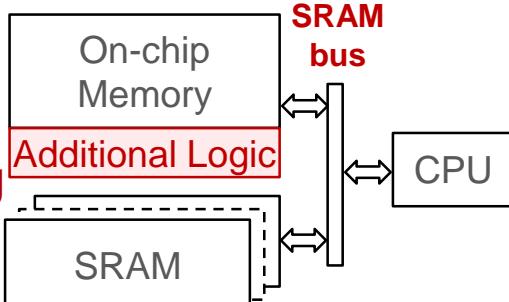
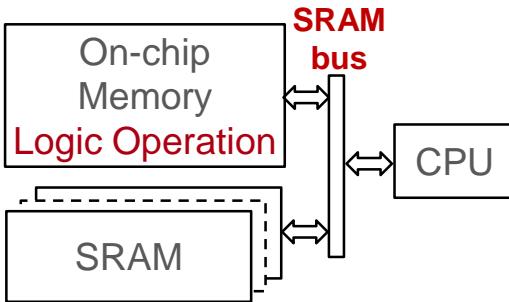
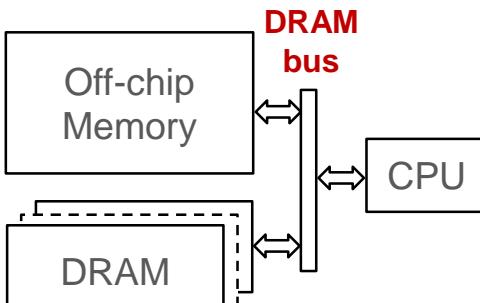
Integer		FP		Memory	
Add		FAdd		Cache	(64bit)
8 bit	0.03pJ	16 bit	0.4pJ	8KB	10pJ
32 bit	0.1pJ	32 bit	0.9pJ	32KB	20pJ
Mult		FMult		1MB	100pJ
8 bit	0.2pJ	16 bit	1.1pJ	DRAM	1.3-2.6nJ
32 bit	3.1pJ	32 bit	3.7pJ		



logic/arith. op. vs memory access energy
→ 100-1000X

- The largest part of power consumption of logic and arithmetic operations is due to the memory access!
 - ✓ The way to perform basic operations has to be restudied
 - ✓ A lot of applications should be improved in performance

DIFFERENT APPROACHES

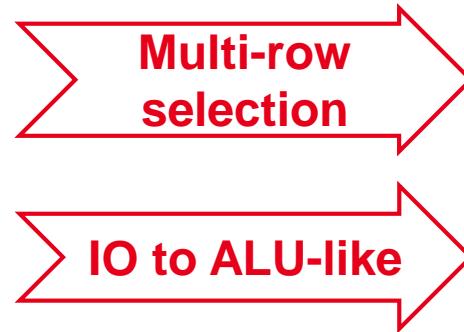
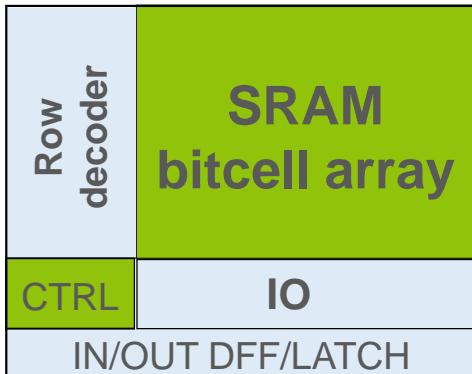
Process	Technique	Description
Embedded Memories (CMOS Process)	In-Memory Computing 	<ul style="list-style-type: none">- Additional logic in memory- Non-destructive computing- Non Volatile/Volatile Memories <p>[Akyel'16] [Aga'17] [Kooli'17]</p>
	Logic-in-Memory 	<ul style="list-style-type: none">- Non-volatile Memories (ReRAM, ...)- Destructive computing <p>[Matsunaga'09]</p>
Stand-alone Memories (DRAM Process)	Processing-in-Memory 	<ul style="list-style-type: none">- Planar / 3D process- Non-destructive computing <p>[Gokhale'95] [Pugsley'14] UpMem</p>

- **Introduction & Context**
- **In-Memory Power Aware ComputIng (IMPACT)**
- **IMPACT Memory Instruction Code**
- **IMPACT Communication Protocol**
- **Conclusion & Perspectives**

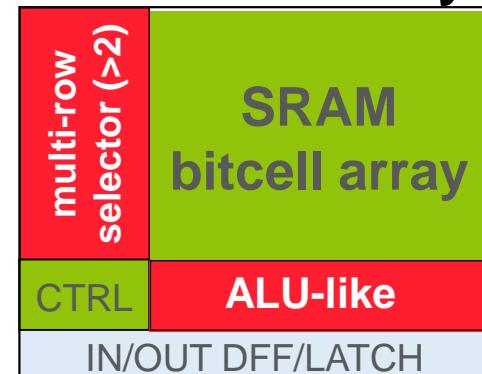
In-Memory Power Aware ComputIng (IMPACT)

- **Computing in dedicated units:**
 - High data transfer between the ALU & the memory
 - Power hungry
 - Interconnect & memory security issues
- **In-memory computing:**
 - Reduced data transfer
 - Energy-efficient
 - Execution time acceleration
 - Security reinforcements (limitation of the side channel attacks (on buses))

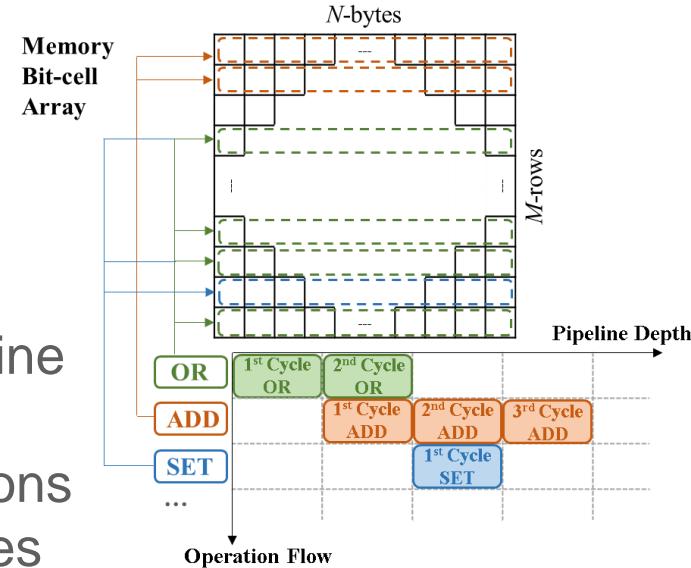
SRAM architecture



IMPACT Memory



- Enable in-memory operations
 - Reduce latency & energy consumption due to data transfer
- SRAM bit cell array allows:
 - Long word arithmetic/logic operations not limited by register size, but with memory line size
 - Multi-row selection for some logic operations
 - Simultaneous storing in different addresses



Emulation Platform:



Emulate the IMPACT system features

- Long word operations
- Multi-operand operations

LLVM

- Early design stage of the system: Not defined ISA
- Manipulate arithmetic/logic operations on large vectors

Target Applications

- Image Processing (*Motion Detection*)
- Cryptography (*One Time Pad*)

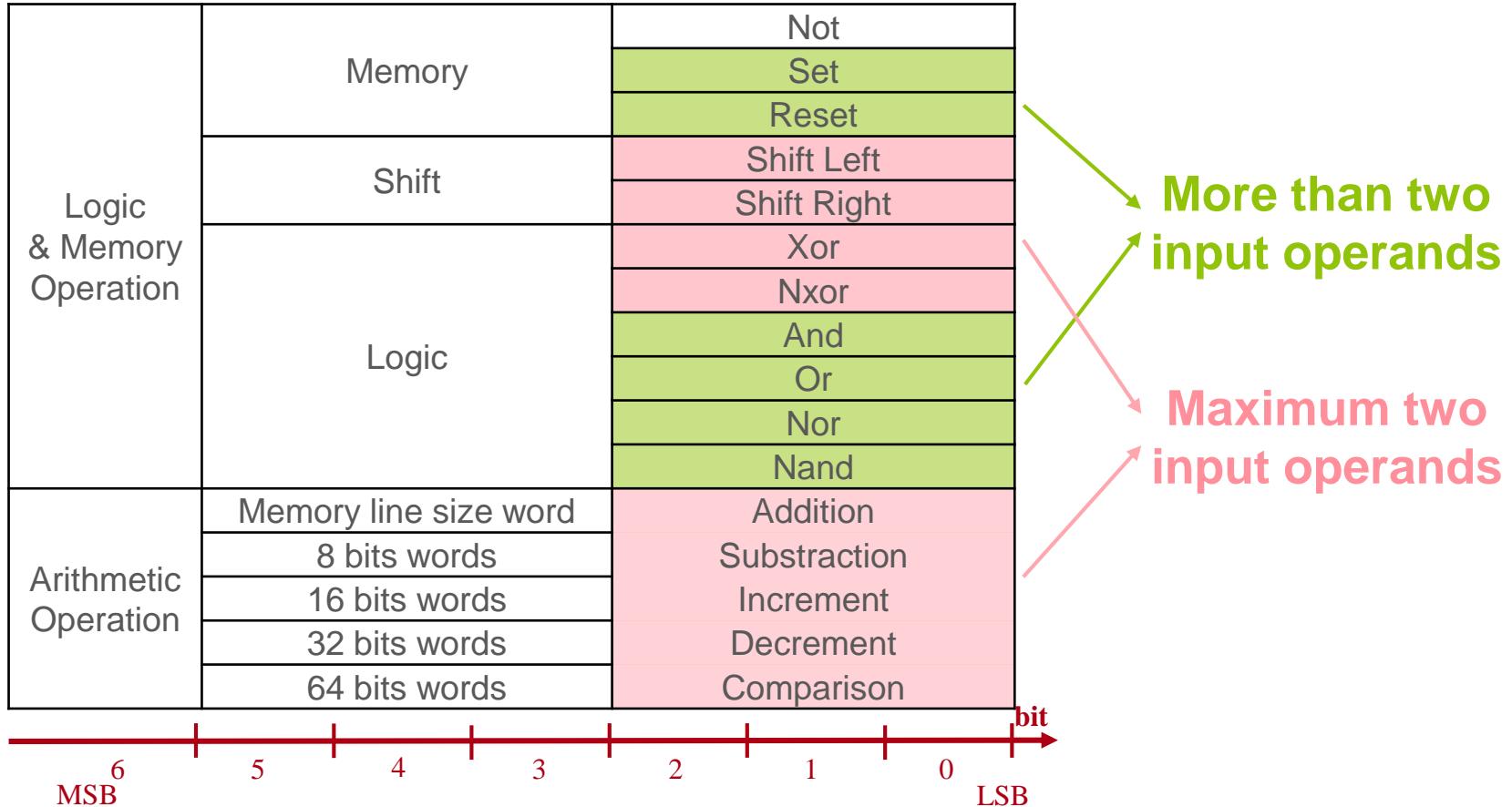
Experimental Gains

- Execution time: up to 6145x
- Energy: up to 12,9x

IMPACT Memory Instruction Code

- Initial idea : put logic operation in bitcells
 - done
- Added idea : add parallel arithmetic in I/O
 - done
- Create an high level emulation platform (based on LLVM)
 - done
- New idea : create an « inverted Von Neuman » protocol (aka ISA)
 - (this presentation) done
- Tape out april 2018
 - on going
- Create a more accurate emulation platform
 - on going
- Create compilation toolbox
 - on going
- Evaluate high level benchmarks
 - on going
- .../...

IMPACT OPERATIONS AKA OPCODES



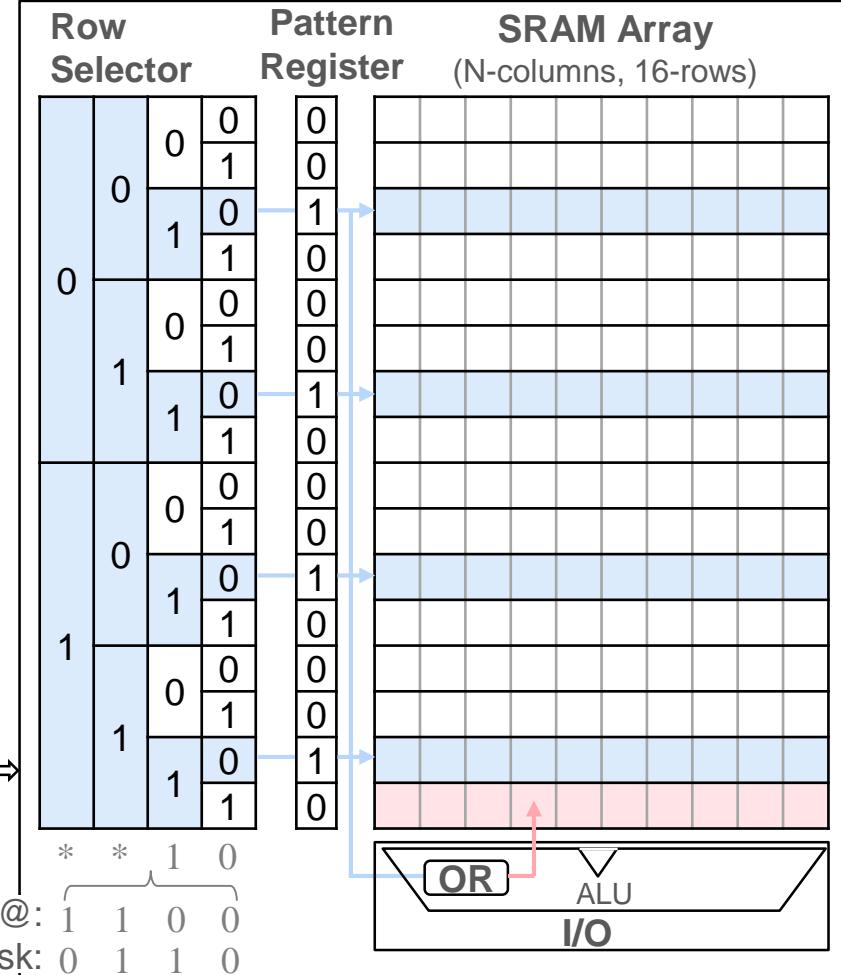
MULTI-OPERAND INSTRUCTION FORMAT

- Multi-operand operations (logic/memory operations)
- Problematic: encoding all the operand addresses in the instruction requires large bus size
 - Propose a novel concept based on **pattern construction**

IMPACT Instruction

Opcode	Pattern Code		SP	Output	SI
	Address	Mask			
OR	0110	1100	1	1111	1

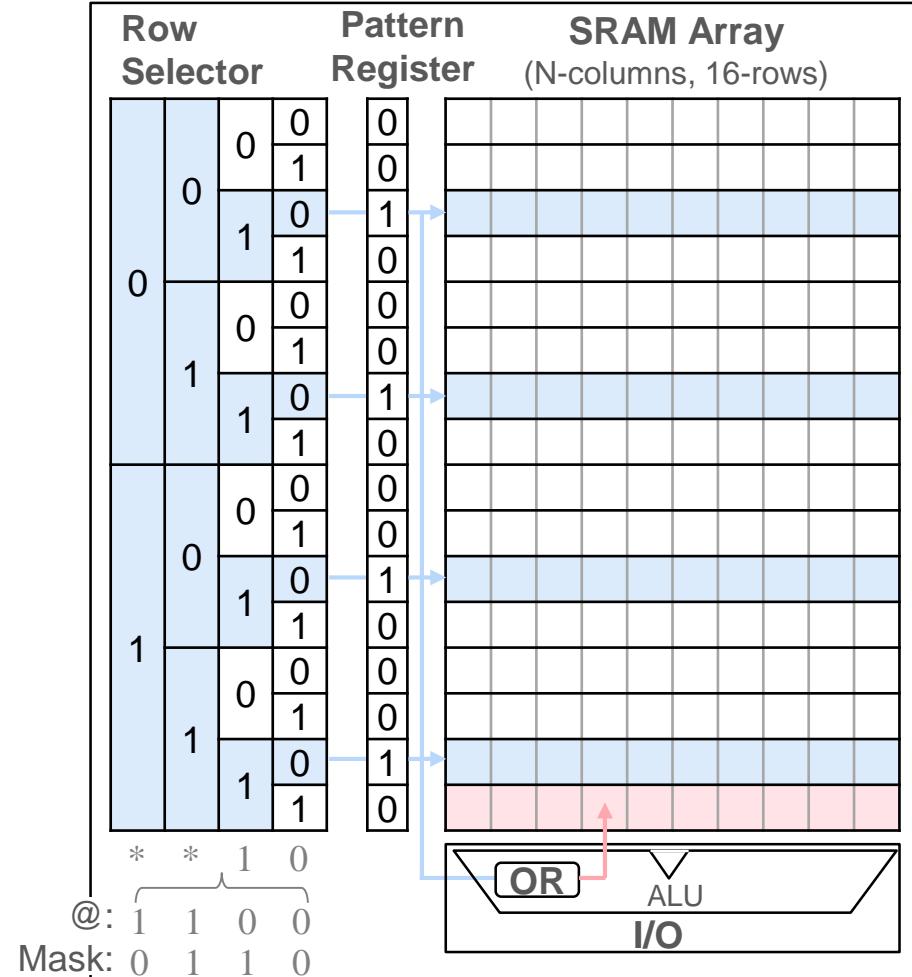
IMPACT Memory



MULTI-OPERAND INSTRUCTION FORMAT

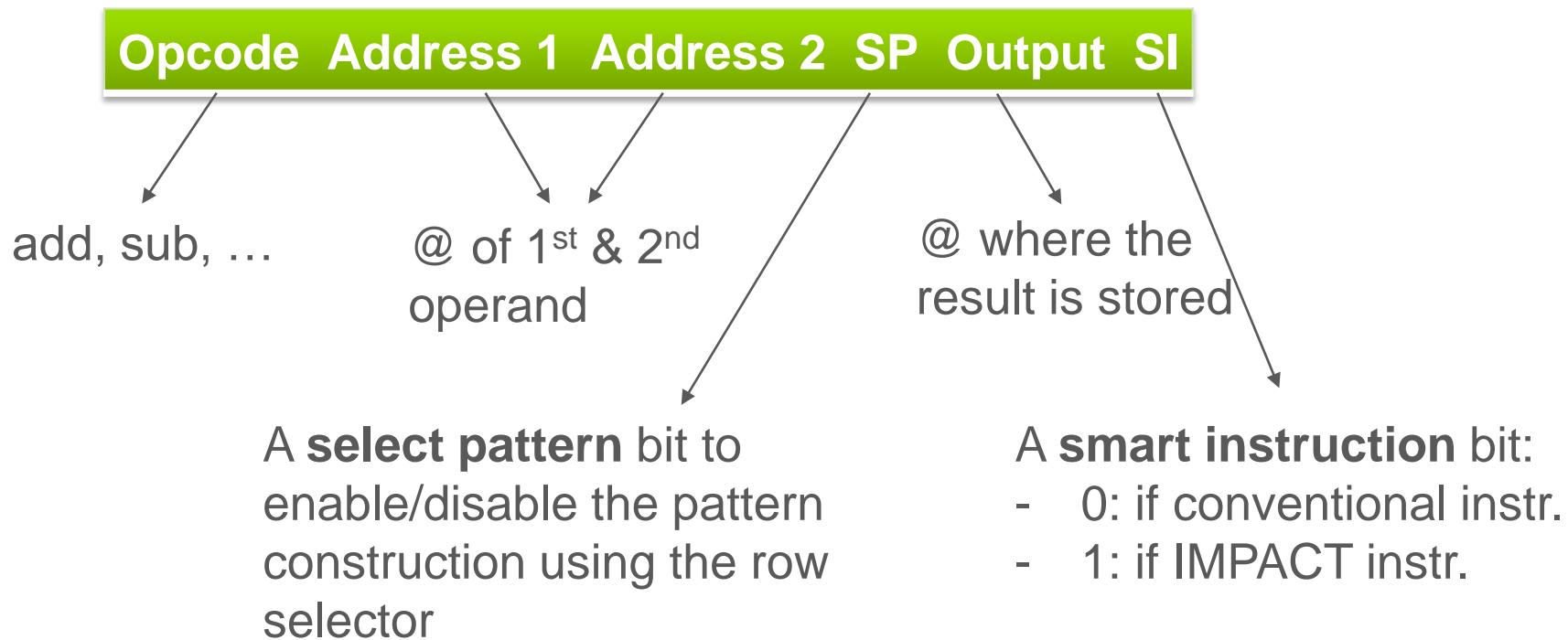
- The proposed method allows:
 - Building regular patterns
 - Patterns can be refined by adding/deleting a specific line
 - Patterns can be stored in the pattern register for future use
 - Selecting multiple lines in the SRAM array to perform the multi-operand operation

IMPACT Memory



TWO-OPERAND INSTRUCTION FORMAT

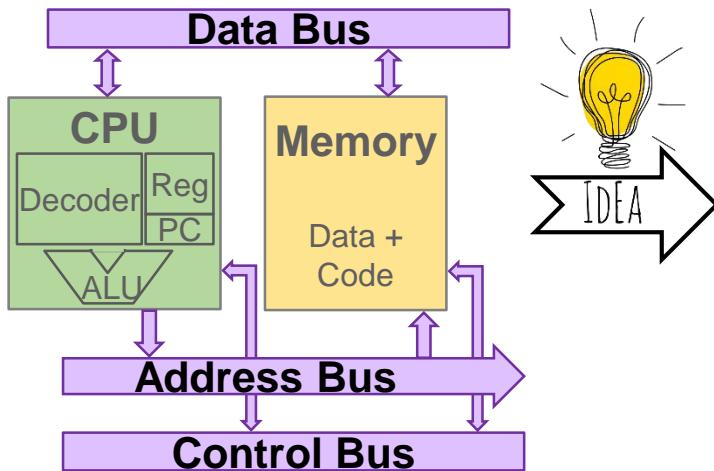
- The conventional format of instruction with maximum two source addresses
- Long-word operations (logic/arithmetic operations)



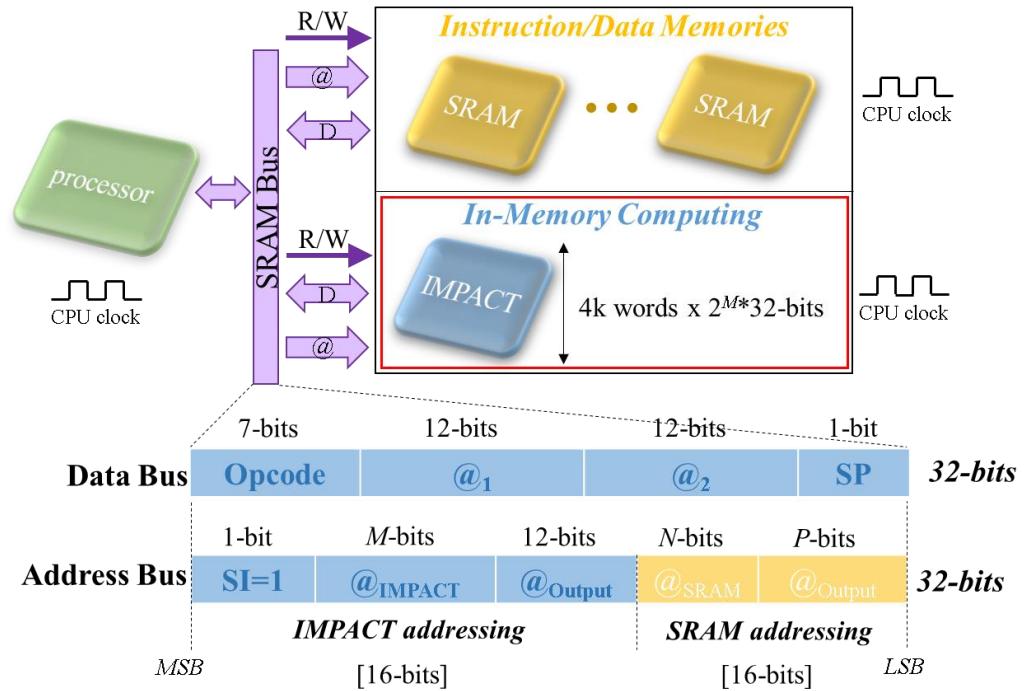
IMPACT Communication Protocol

COMMUNICATION PROTOCOL

Conventional System



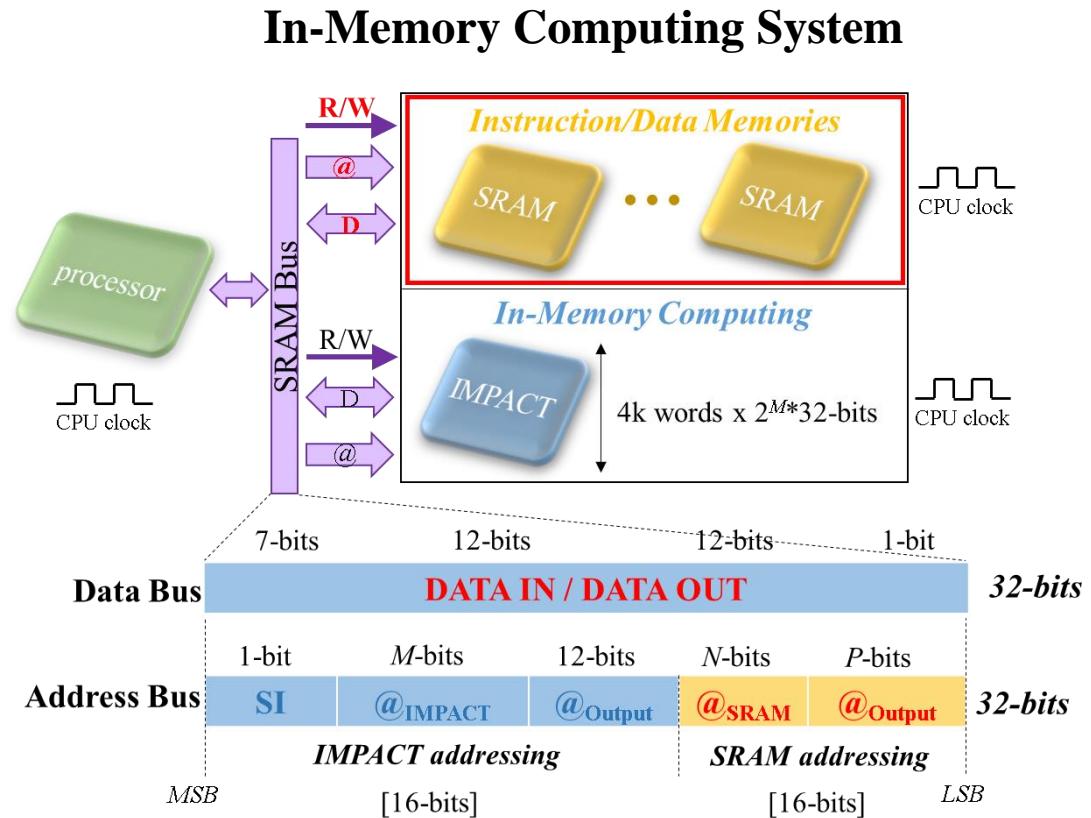
In-Memory Computing System



- Communicate in-memory instructions via **data & address busses** of a conventional system
 - Compatible with existing system **architecture** (*conventional system bus*)
 - Enable interleaving the **CPU** & the **in-memory** instruction execution

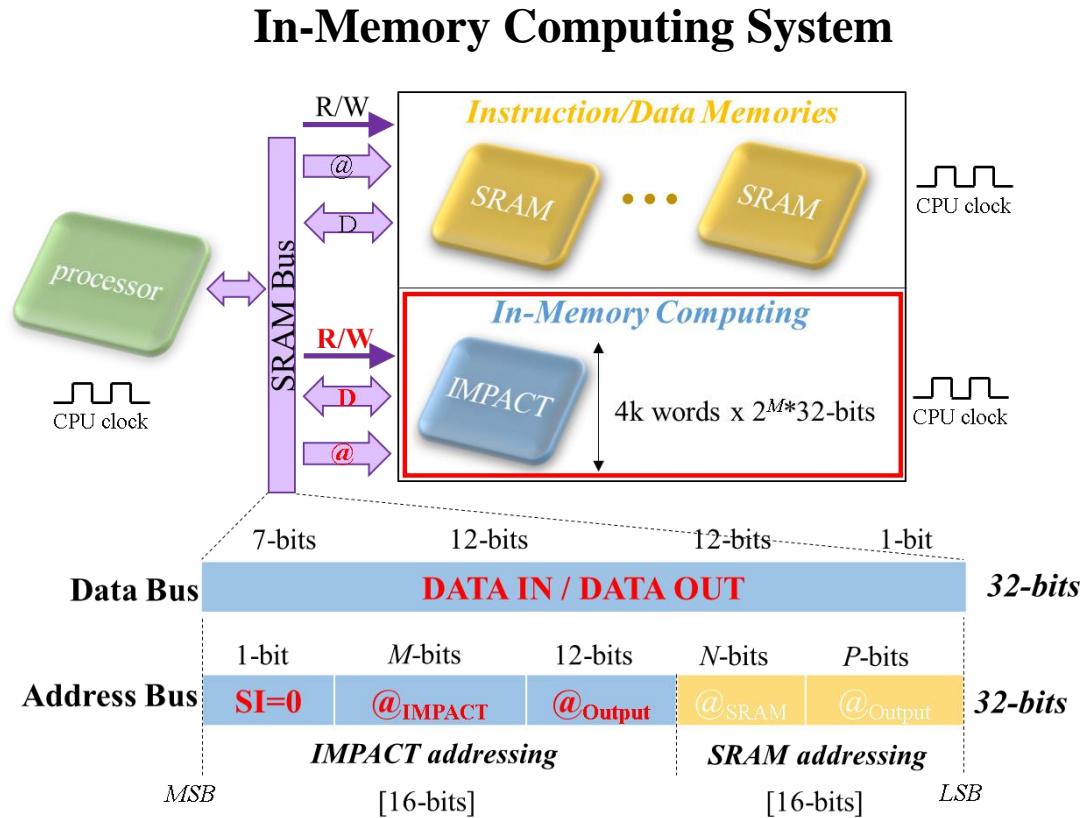
COMMUNICATION PROTOCOL

1. Address the SRAM in conventional mode

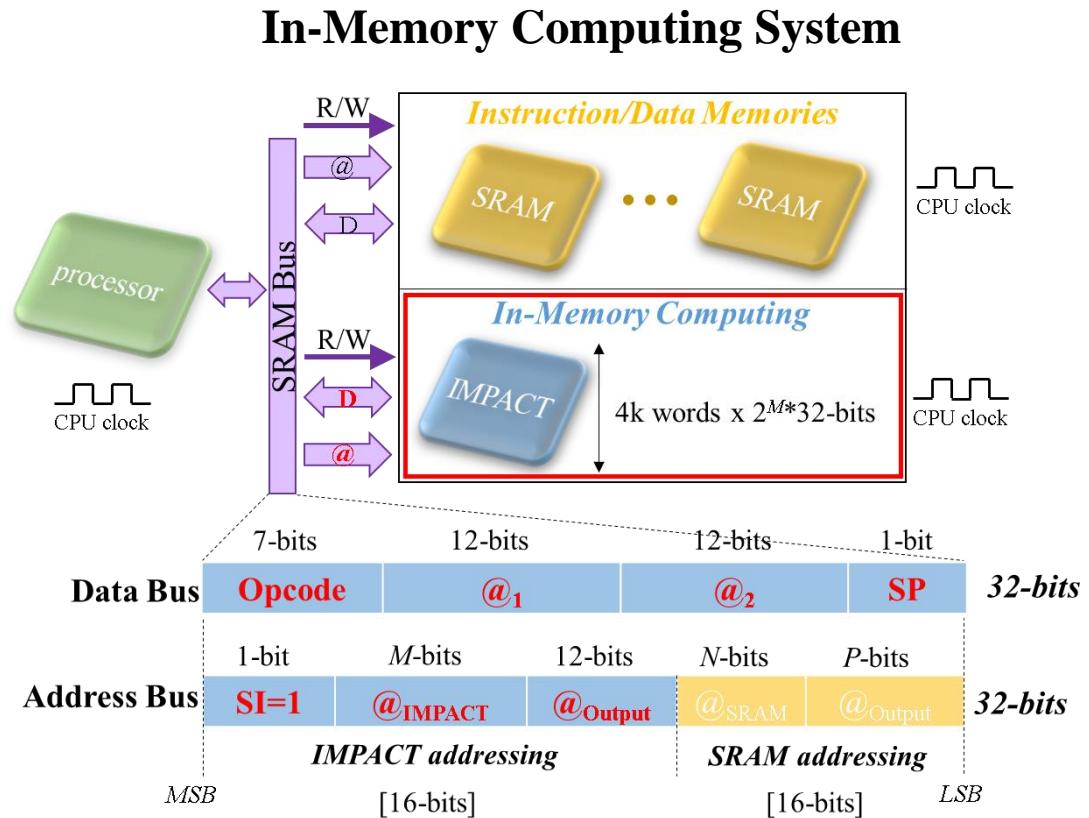


COMMUNICATION PROTOCOL

2. Address the IMPACT memory for read/write



3. Address the IMPACT memory for computation



DATA INTERLEAVING: A TEST CASE

IMC Application

```
typedef unsigned char ImgLine_attribute_((ext_vector_type(N)));
typedef ImgLine Image[M];

for (line = 0; line < M; ++line){
    Img[line] = imgA[line]-imgB[line];
}
```

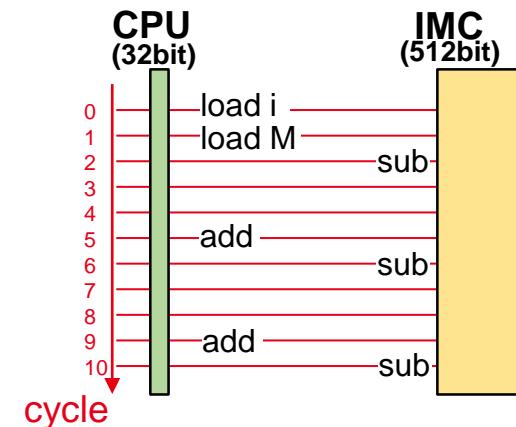


Execution Trace

sub	512	0x7d640	0x7d240	0x7ce40
add	32	0x00001		
sub	512	0x7e640	0x7e240	0x7cf40
add	32	0x00001		
...				

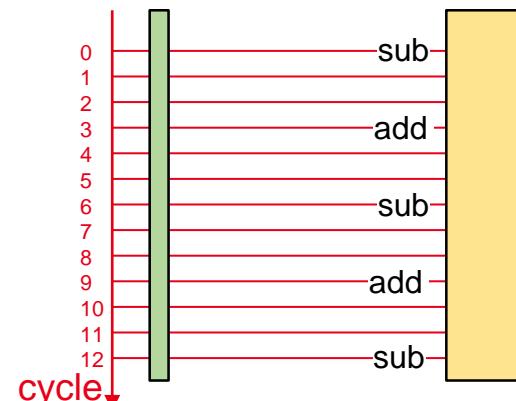
1. Interleave CPU & IMC instruction execution

- Perform massive data computation inside IMC, and not optimized computation in CPU
 - For image **qqVGA 160x120 (not pipelined)**:
 - Execution Time Speed-Up: **1376x**
 - Energy Reduction Factor: **29x**



2. Perform all the computation inside IMC

- For image **qqVGA 160x120 (not pipelined)**:
 - Execution Time Speed-Up: **765x**
 - Energy Reduction Factor: **29x**



COMMUNICATION PROTOCOL INSTRUCTION SET ARCHITECTURE

Source Code

```
...  
R = s1 + s2  
...
```

Compilation

Assembly Code

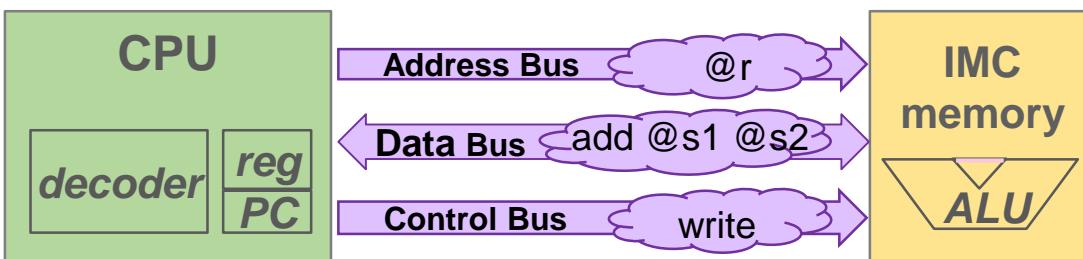
```
...  
mv      r2, @R  
mv      r1, #add  
shl    r1, #6  
xor    r1, #@s1  
shl    r1, #6  
xor    r1, #@s2  
store r1, r2  
...
```

**Prepare
registers**

**Perform IMC
operation**

IMC Instruction:

add	@s1	@s2	@R
-----	-----	-----	----



- + Do not change the actual ISA
- Overhead preparation

Architecture scenario w/o ISA modification

Conclusion & Perspectives

CONCLUSION & PERSPECTIVES

- Propose a new communication protocol between the CPU and the IMPACT memory
 - Compatible with existing system **architecture** (*conventional system bus*)
 - Enable interleaving the **CPU** & the **in-memory** instruction execution
- Work on the compiler:
 - Generate the assembly code respecting the communication protocol
 - Interleave the IMC & CPU instruction execution
 - Based on the performance evaluation
 - Optimize the data set-up in the memory
 - Data alignment in the IMC
 - Data interleaving

Leti, technology research institute
Commissariat à l'énergie atomique et aux énergies alternatives
Minatec Campus | 17 rue des Martyrs | 38054 Grenoble Cedex | France
www.leti.fr

