Processing in Memory Execution Targets for Higher Level Languages

David Krasowska
Ph.D. Student

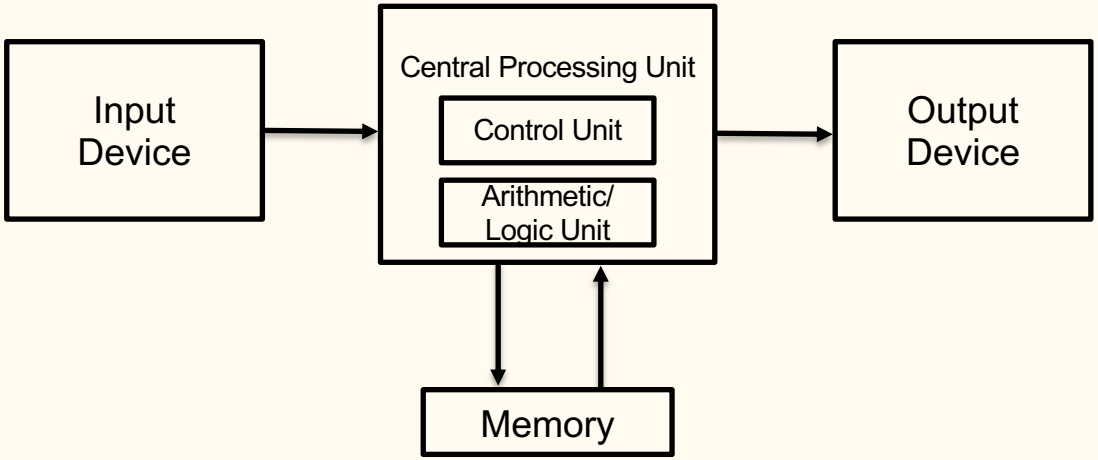
Northwestern | ENGINEERING

Constellation Project Workshop

Evanston, IL

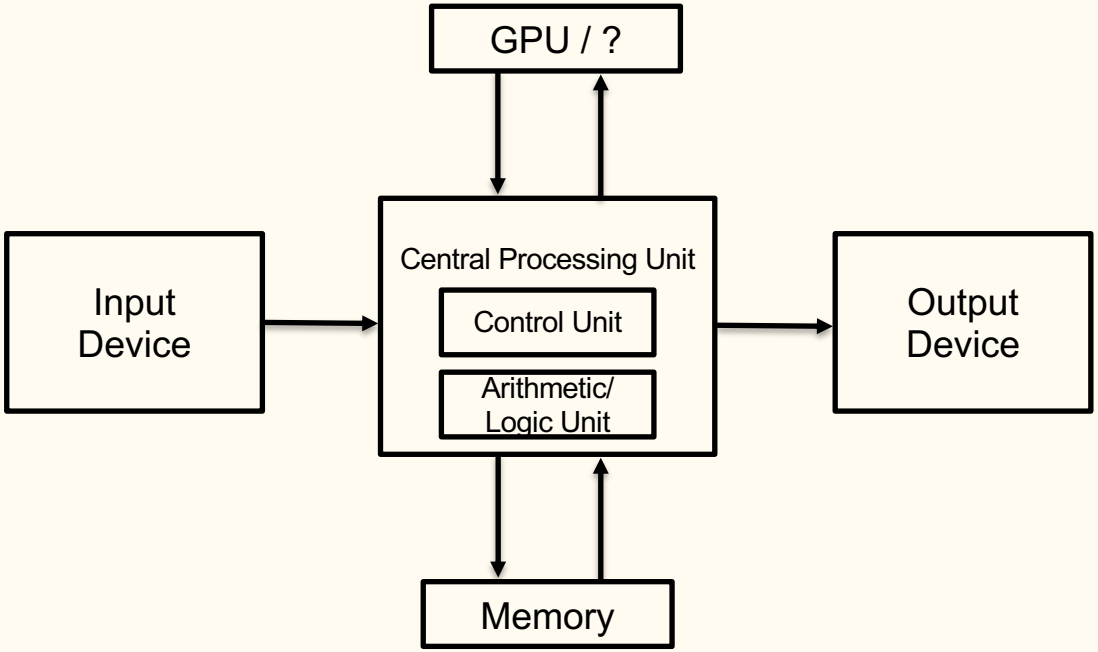
July 24, 2023

Compute centric architecture



https://en.wikipedia.org/wiki/Von_Neumann_architecture

Let’s add an accelerator



Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

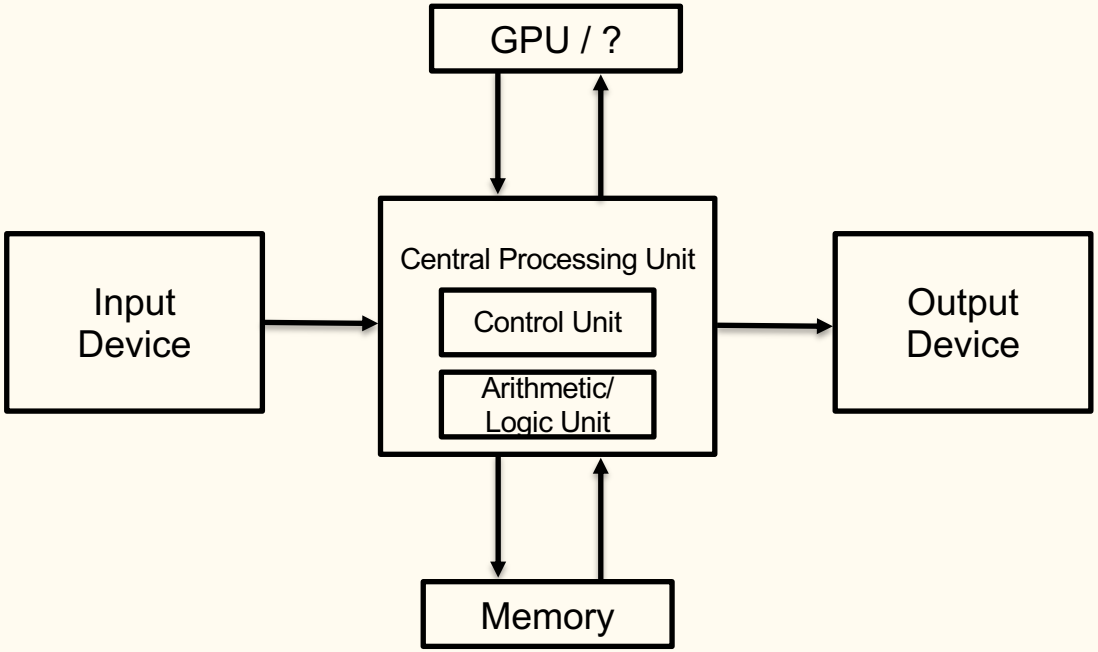
Amirali Boroumand[1] Saugata Ghose[1] Youngsok Kim[2]
Rachata Ausavarungnirun[1] Eric Shiu[3] Rahul Thakur[3] Daehyun Kim[4,3]
Aki Kuusela[3] Allan Knies[3] Parthasarathy Ranganathan[3] Onur Mutlu[5,1]

[1]Carnegie Mellon University [2]Dept. of ECE, Seoul National University [3]Google [4]Samsung Research [5]ETH Zürich

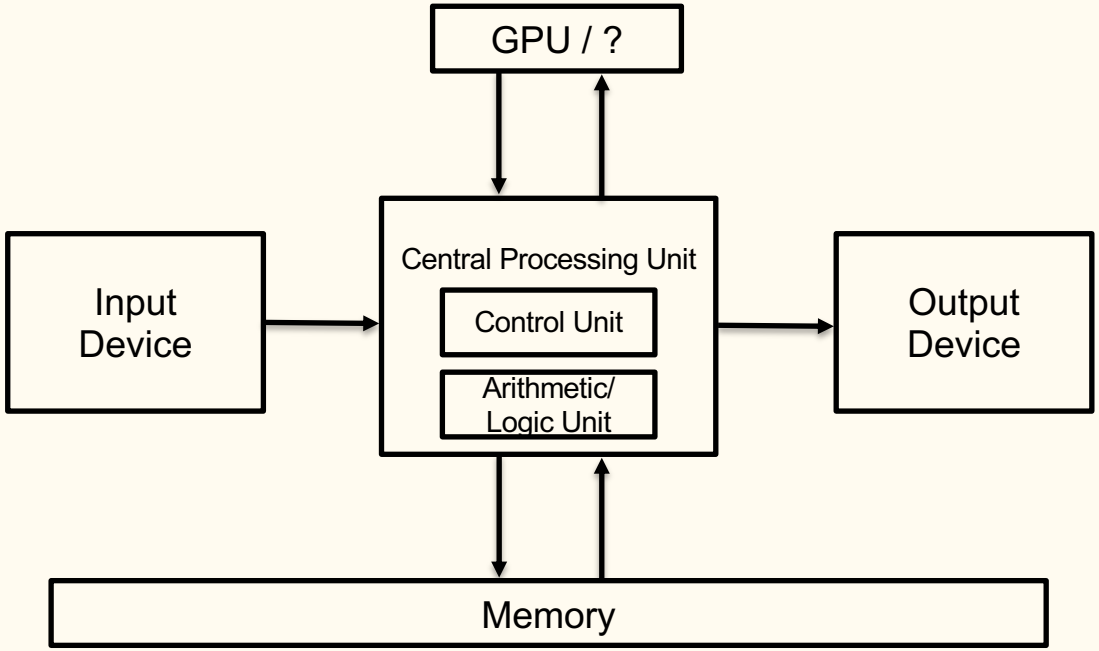
~60% of energy on consumer workloads is used on data movement

https://www.pdl.cmu.edu/PDL-FTP/associated/asplos18-pim-final.pdf

What if we expanded what memory does?

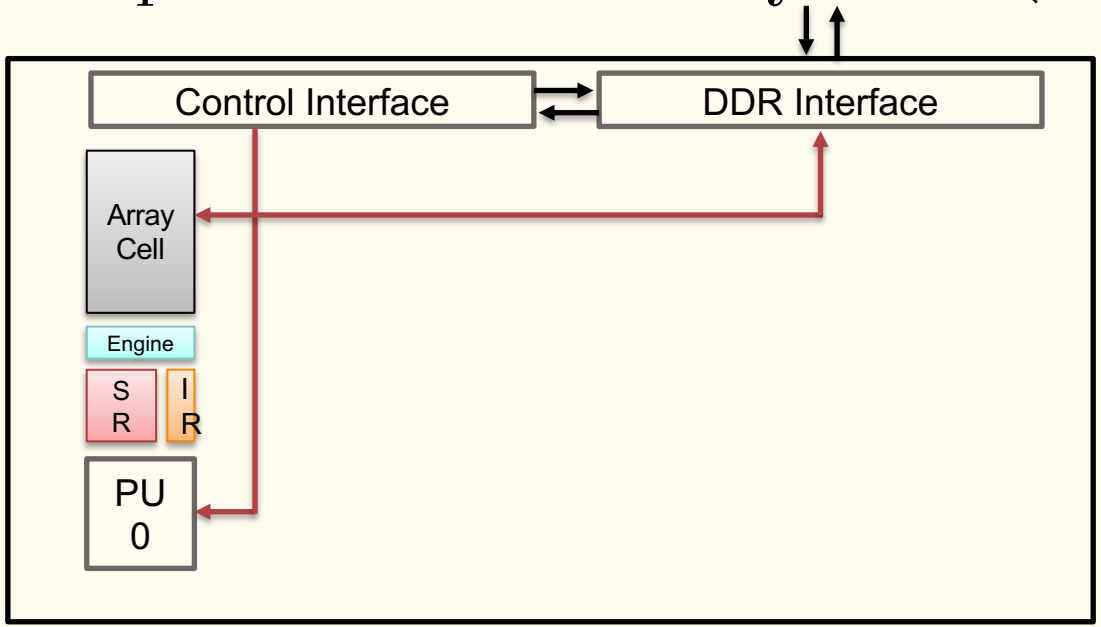


What if we expanded what memory does?



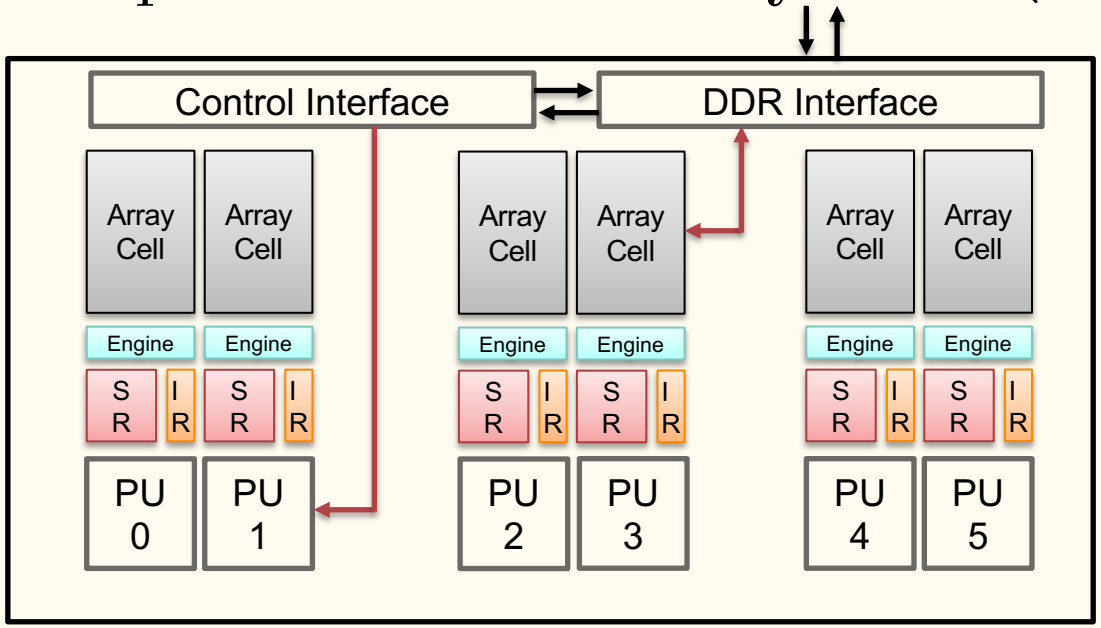
What if we expanded what memory does? (example)

Memory
(expanded view)



What if we expanded what memory does? (example)

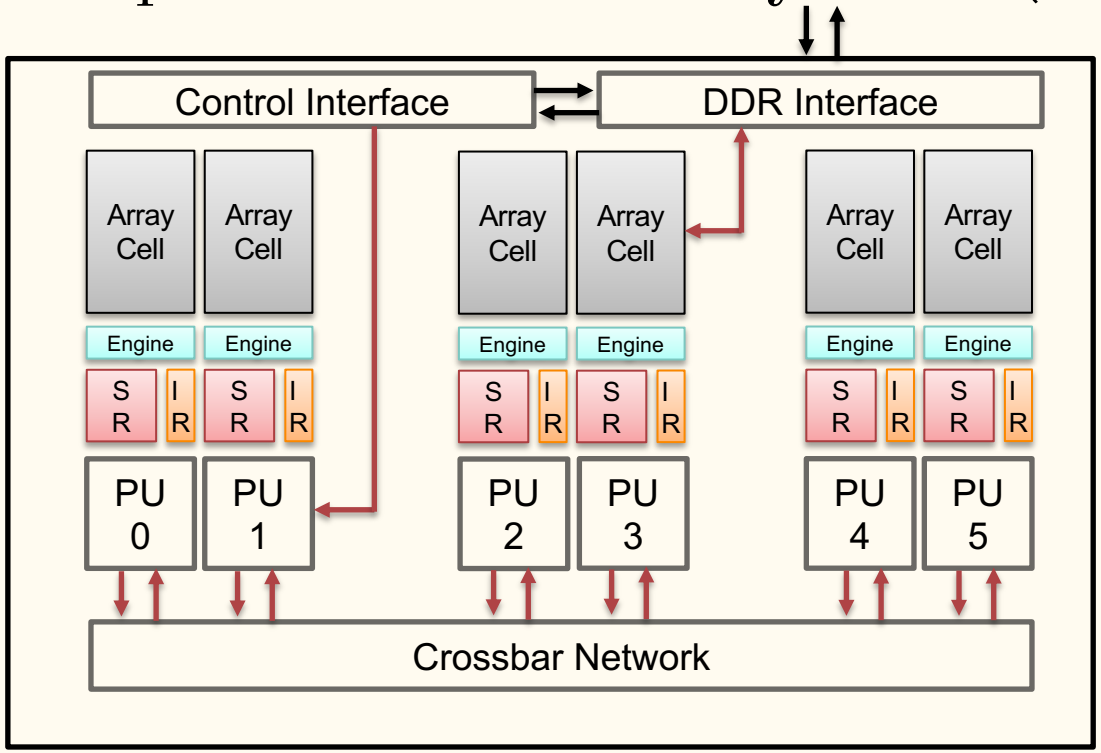
Memory
(expanded view)



How do different **PUs** communicate with one another?

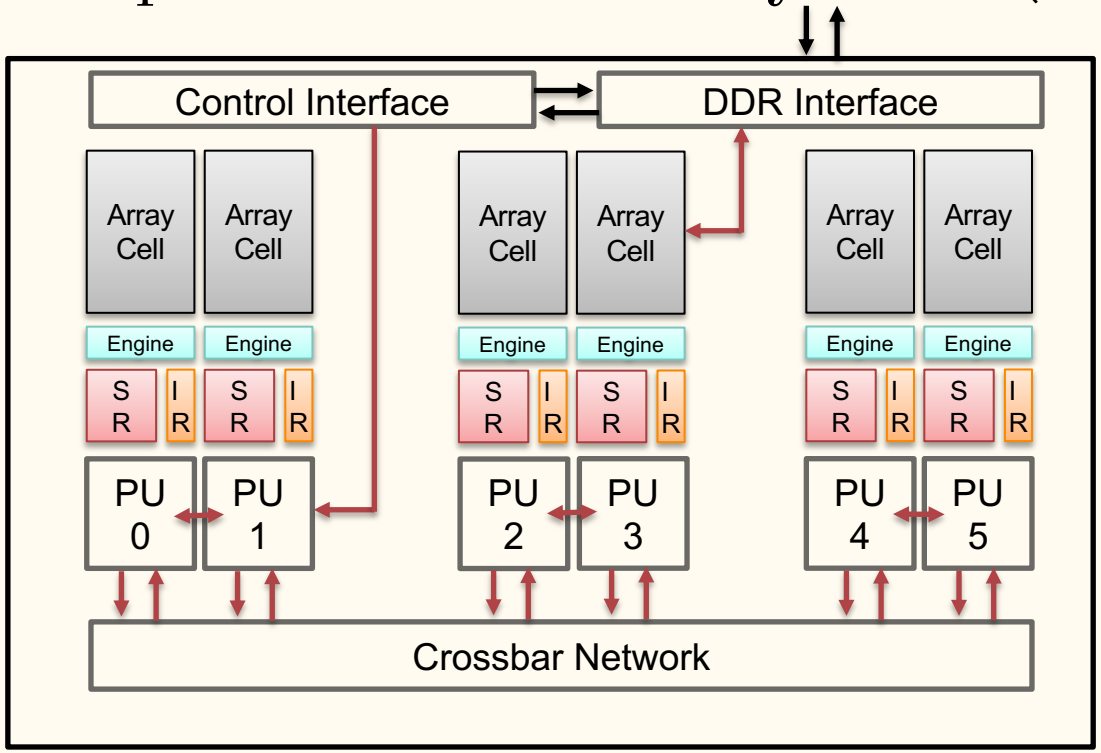
What if we expanded what memory does? (example)

Memory
(expanded view)



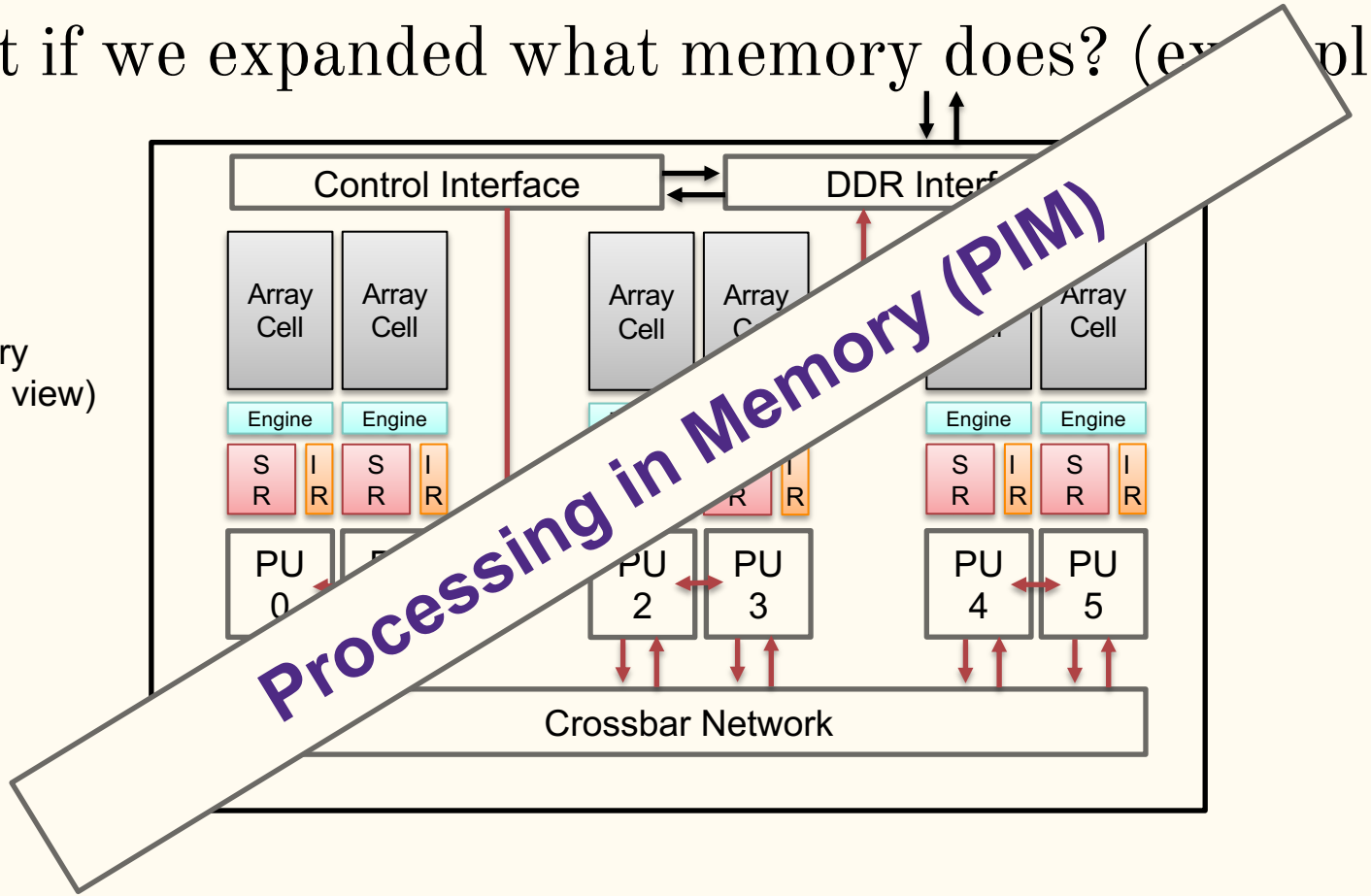
What if we expanded what memory does? (example)

Memory
(expanded view)



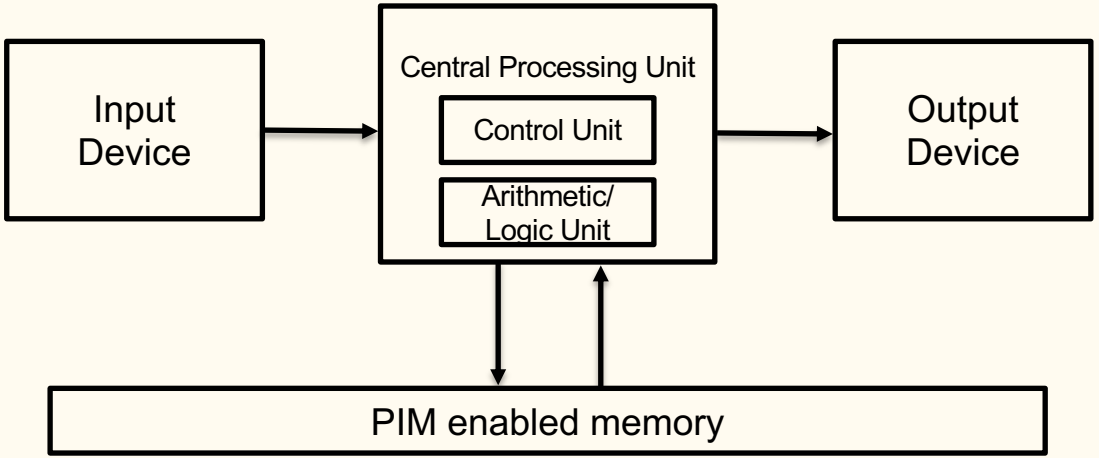
What if we expanded what memory does? (ex ple)

Memory
(expanded view)



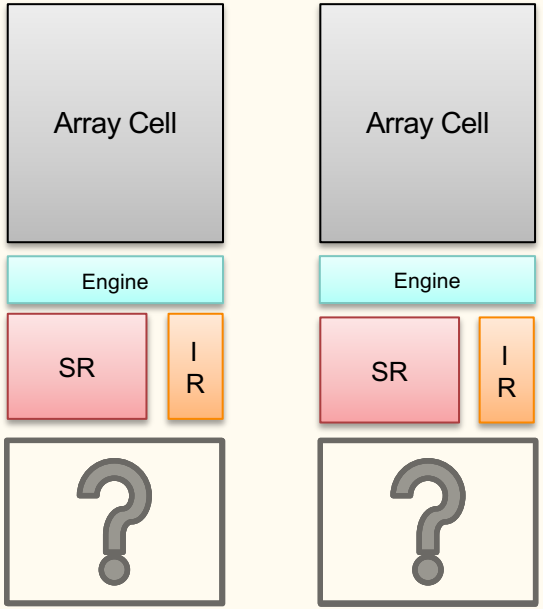
Processing in Memory (PIM)

- Computation on DRAM to perform instructions
- Interrupting the typical compute centric architecture
- Reduces the data movement bottleneck



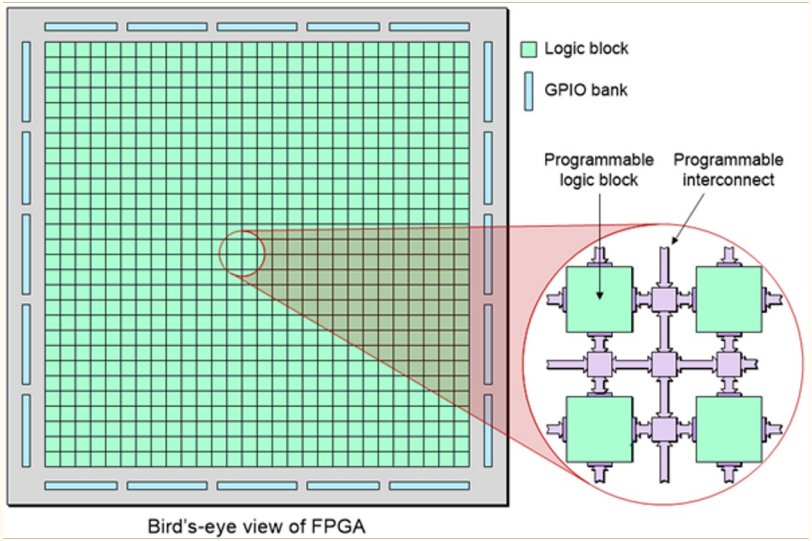
Lots of factors to consider...

- Supported ISA operations?
- Communication network?
- How many pipeline stages?
- Hardware threads?
- Memory sizes (instruction, scratchpad, main)?

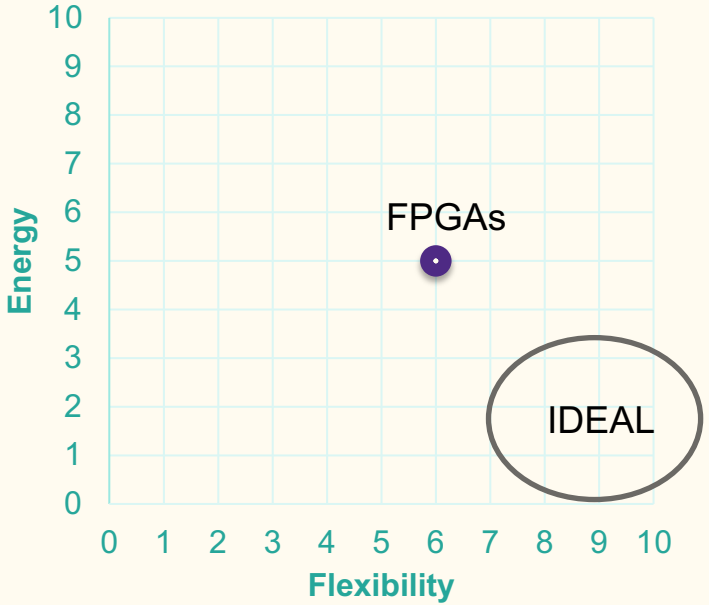


What is inside a PU?

FPGA?

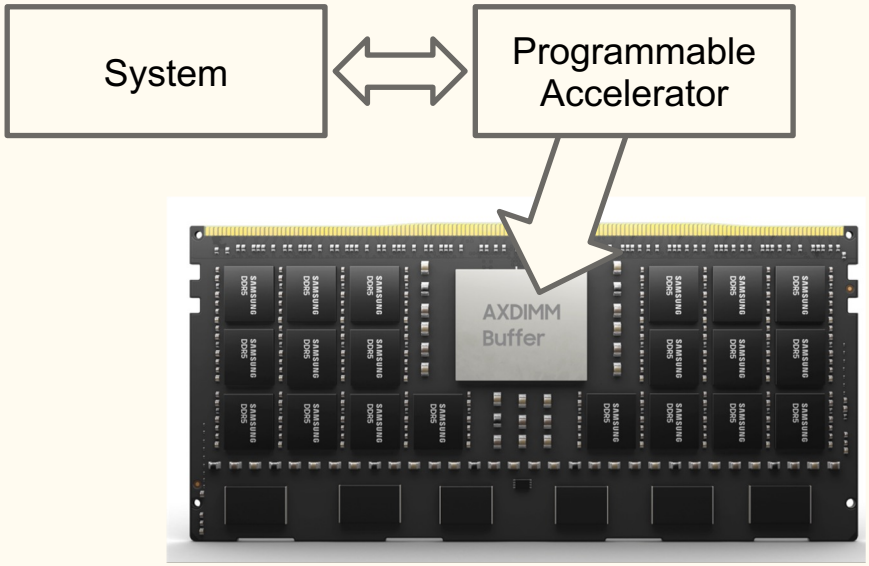


FPGAs

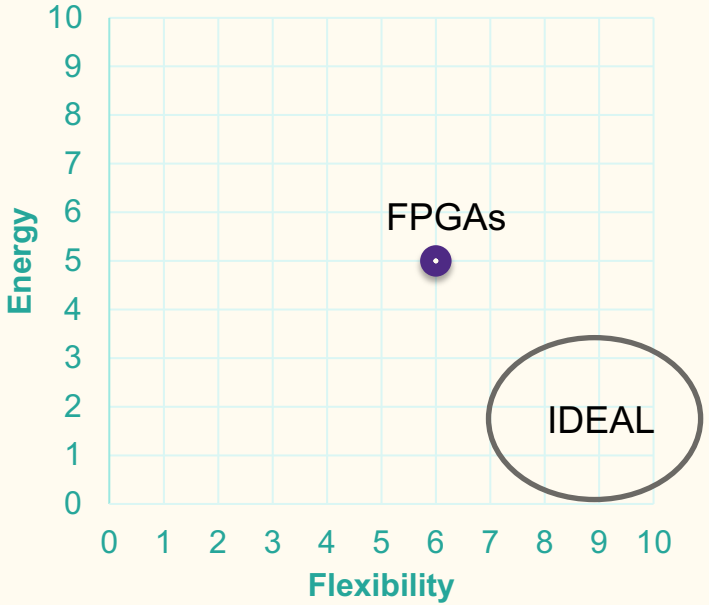


https://www.latticesemi.com/en/What-is-an-FPGA

FPGA?

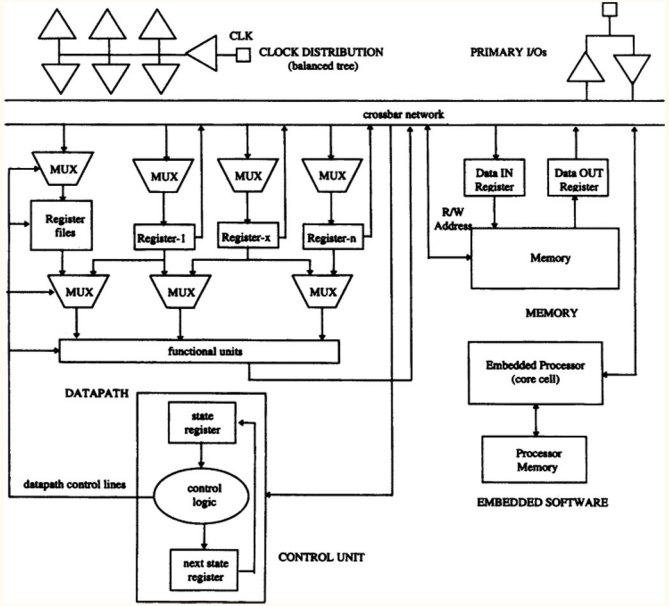


FPGAs

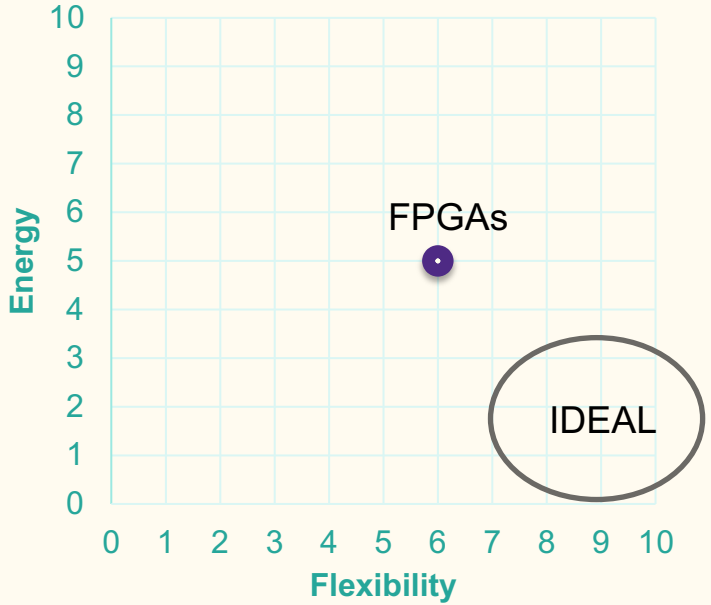


https://news.semiconductorsamsung.com

ASIC?

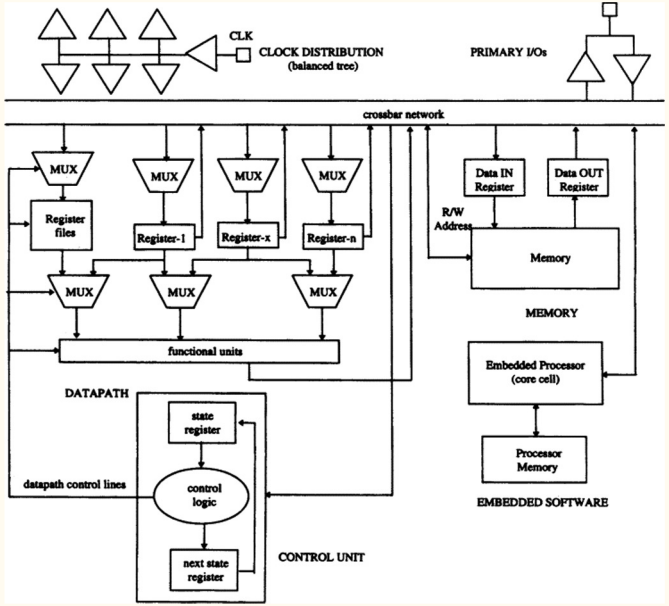


ASICs

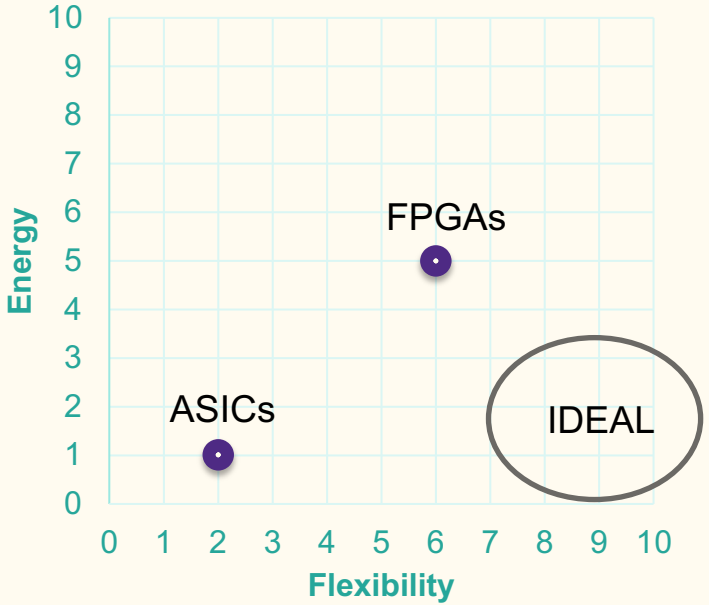


https://doi.org/10.1016/B978-155860702-6/50021-1

ASIC?

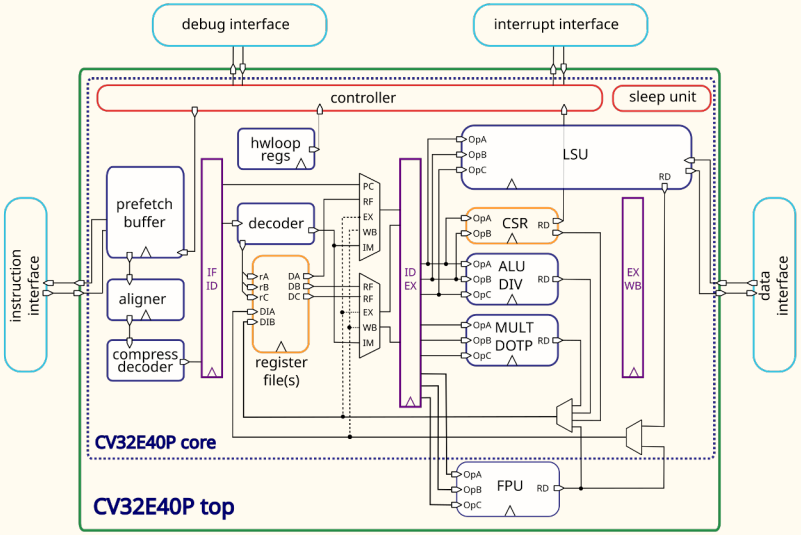


ASICs

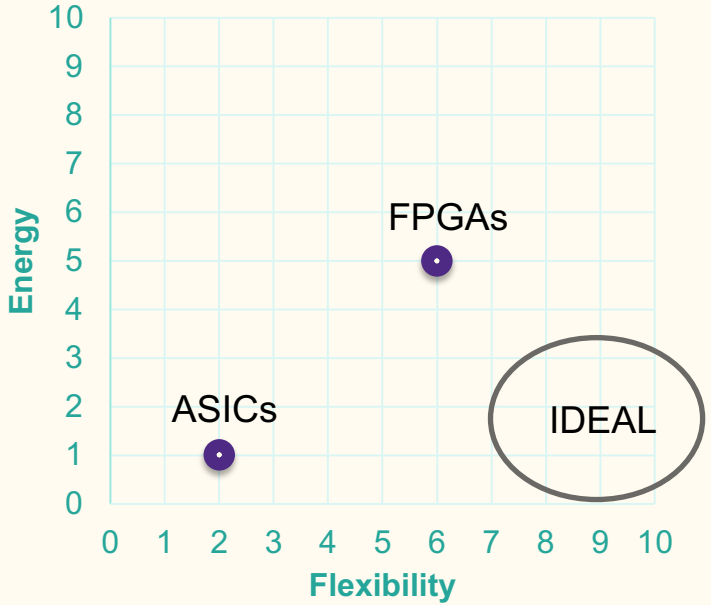


https://doi.org/10.1016/B978-155860702-6/50021-1

RISC style general purpose processor?

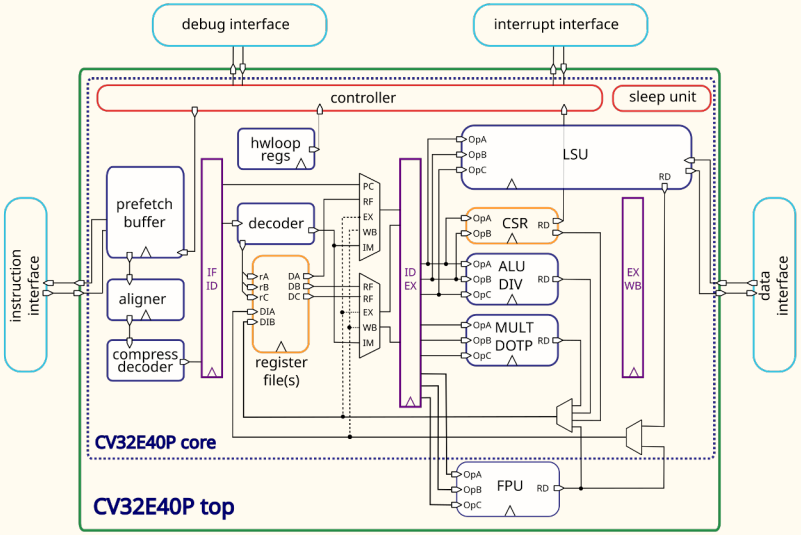


General Purpose Processor

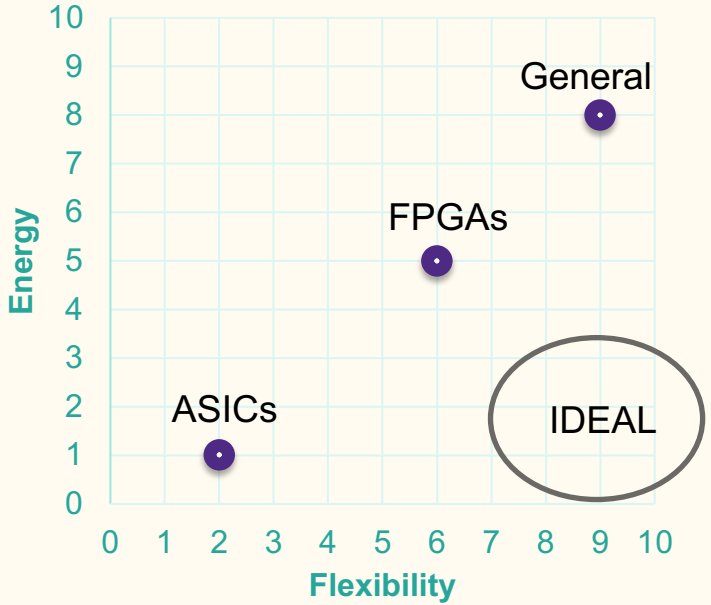


https://docs.openhwgroup.org/projects/cv32e40p-user-manual/en/latest/intro.html

RISC style general purpose processor?



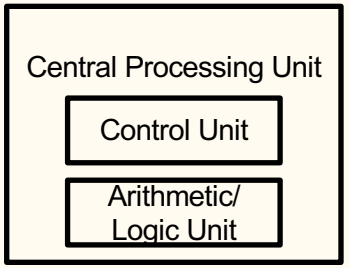
General Purpose Processor



https://docs.openhwgroup.org/projects/cv32e40p-user-manual/en/latest/intro.html

There are multiple available architectures

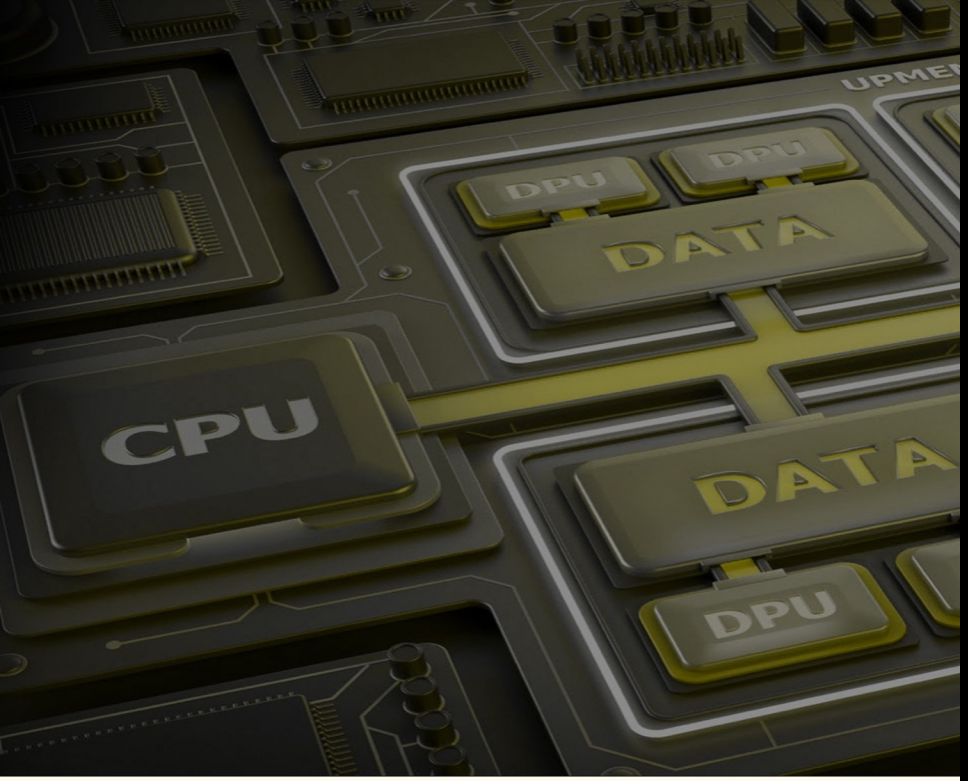
- UPMEM, SAMSUNG, SK Hynix, Alibaba ... AND MORE
- All require complex central CPU
  - Manage the processing units
  - Manage I/O
- Utilize a SPMD execution model



UPMEM DIMMs

- State of the art general purpose PIM architecture
- SDK with a simulator
- Requires HOST TO DPU transfers to execute kernels
- Inter-DPU communication utilizes the CPU

https://www.upmem.com

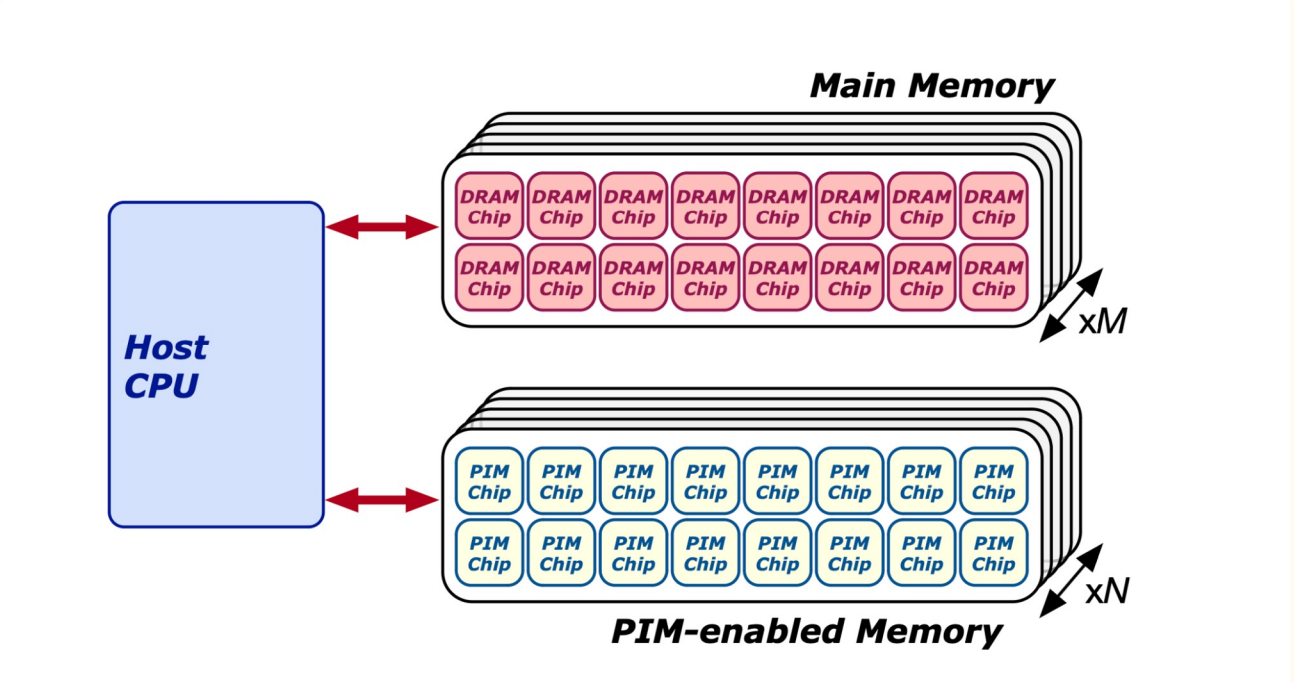


Hardware configuration

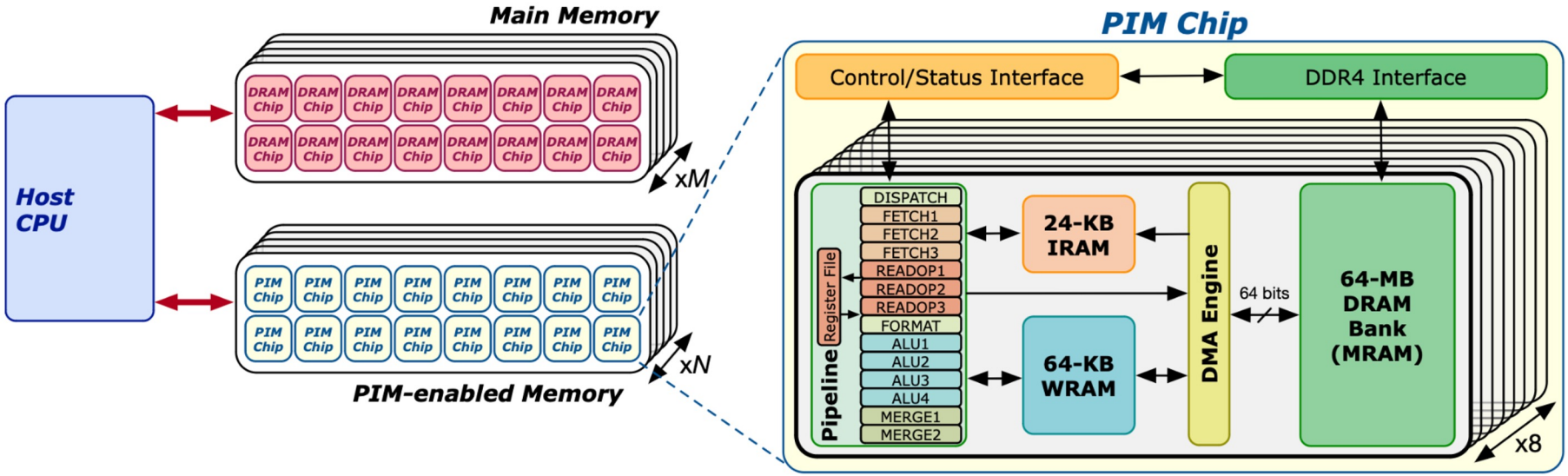
- UPMEM PIM DIMM w/128 DPUs
- 450Mhz (soon 600 Mhz)
- DPU has 24 threads with 32-bit RISC processor (64-bit capabilities)
- 2560 DPUs for a single socket server with 256GB PIM DRAM

https://www.upmem.com

Hardware configuration



Hardware configuration



ISA for UPMEM

Operation	Description
=	Set the value of a register (or temporary variable
<-	Set the value of one or multiple flags
jump	Change PC value
call	Save the value of PC+1 and change PC value
boot	Reset the PC of a thread and start it
resume	Start a thread without resetting its PC value
acquire	Acquire a lock
release	Release a lock
read RB	Read the specified run bit
clear RB	Clear the specified run bit
T:config	Configure the performance counter
T:read	Read the performance counter
sats	Saturation function: (value < 0) ? 0x7FFFFFFF : 0x80000000

~	Flip bits
+	Arithmetic add
-	Arithmetic sub
*	Multiply
|	Bitwise or
&	Bitwise and
^	Bitwise exclusive or
>>	Shift to the right
<<	Shift to the left
<<r	Rotate bits to the left
>>r	Rotate bits to the right
>>a	Arithmetic shift to the right

- Multiply is 8 bits only

Programming on UPMEM

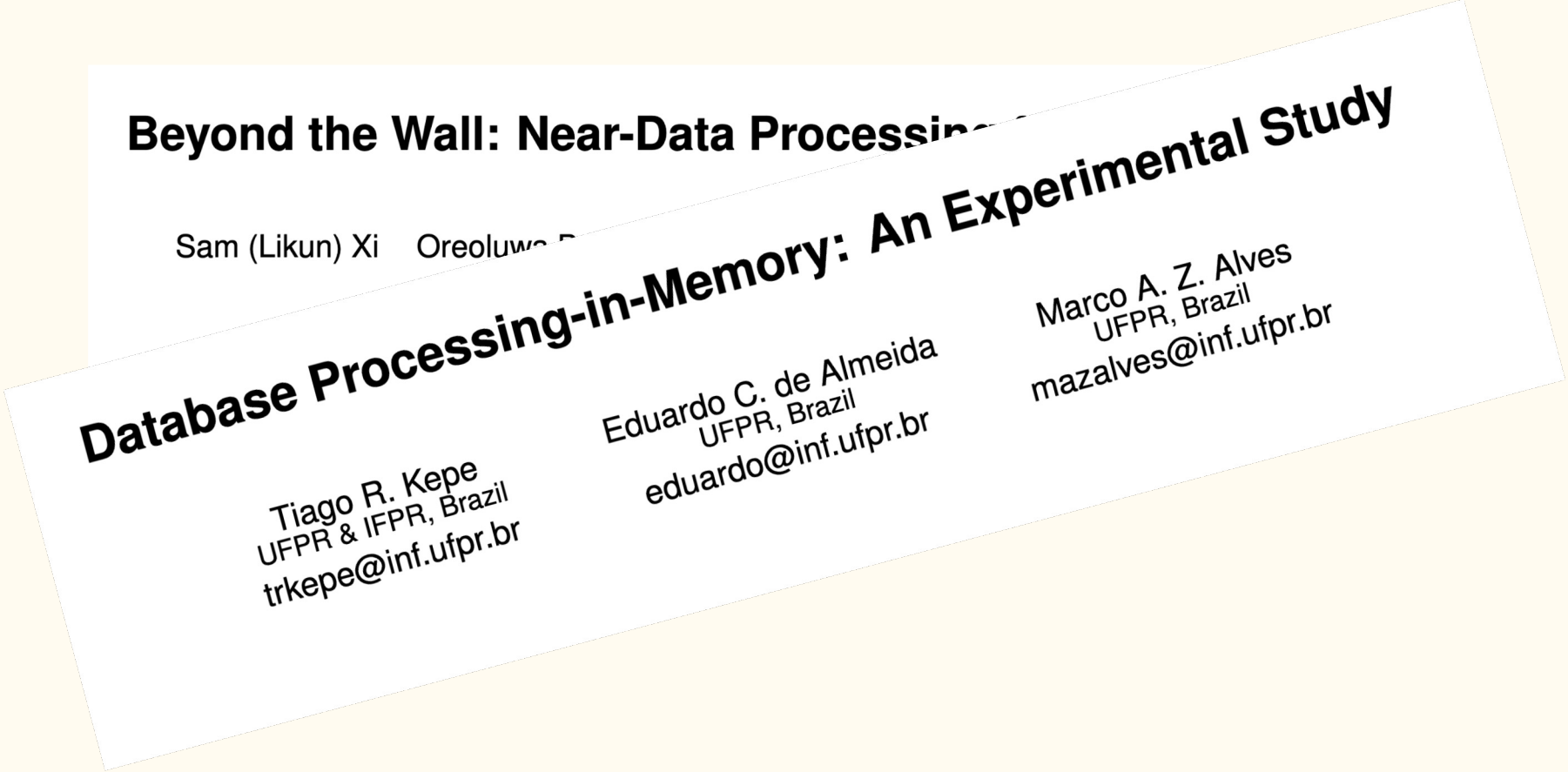
1. Execute parallel code as long as possible
2. Split data into independent blocks
3. Use as many working DPUs as possible
4. Reduce inter and intra DPU synchronization

Beyond the Wall: Near-Data Processing for Databases

Sam (Likun) Xi Oreoluwa Babarinsa Manos Athanassoulis Stratos Idreos

Harvard University

{samxi, obabarinsa, manos, stratos}@seas.harvard.edu



Beyond the Wall: Near-Data Processing

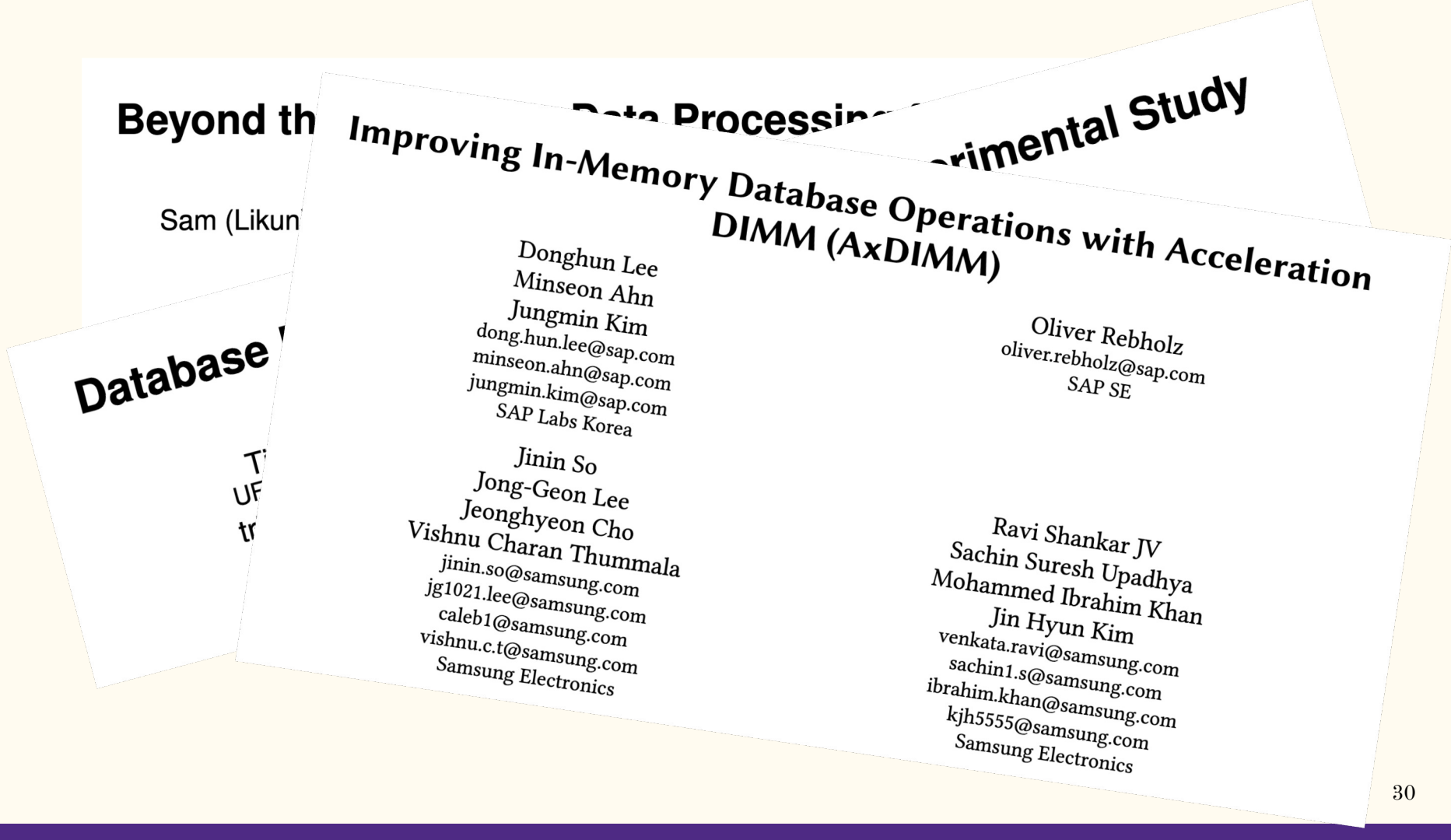
Sam (Likun) Xi Oreoluwa

Database Processing-in-Memory: An Experimental Study

Tiago R. Kepe
UFPR & IFPR, Brazil
trkepe@inf.ufpr.br

Eduardo C. de Almeida
UFPR, Brazil
eduardo@inf.ufpr.br

Marco A. Z. Alves
UFPR, Brazil
mazalves@inf.ufpr.br


Beyond th
Sam (Likun
Database
Improving In-Memory Database Operations with Acceleration DIMM (AxDIMM)
Donghun Lee
Minseon Ahn
Jungmin Kim
dong.hun.lee@sap.com
minseon.ahn@sap.com
jungmin.kim@sap.com
SAP Labs Korea
Jinin So
Jong-Geon Lee
Jeonghyeon Cho
Vishnu Charan Thummala
jinin.so@samsung.com
jg1021.lee@samsung.com
caleb1@samsung.com
vishnu.c.t@samsung.com
Samsung Electronics
Oliver Rebholz
oliver.rebholz@sap.com
SAP SE
Ravi Shankar JV
Sachin Suresh Upadhya
Mohammed Ibrahim Khan
Jin Hyun Kim
venkata.ravi@samsung.com
sachin1.s@samsung.com
ibrahim.khan@samsung.com
kjh5555@samsung.com
Samsung Electronics

What is SQL?

PostgreSQL

- Typical query plan description language utilized by multiple DBMS
- Based on many relational algebra concepts (joins, predicates, etc.)
- Can be used for both transactional and analytical workload
- High level language benefits

TPCH Query 6

```
select
        sum(l_extendedprice * l_discount) as revenue
from
        lineitem
where
        l_shipdate >= date '1994-01-01'
        and l_shipdate < date '1995-01-01'
        and l_discount between 0.06 - 0.01 and 0.06 + 0.01
        and l_quantity < 24
```

TPCH Q6 lineitem table schema

```
CREATE TABLE lineitem
(
        l_orderkey            BIGINT not null,
        l_partkey             BIGINT not null,
        l_suppkey             BIGINT not null,
        l_linenumber          BIGINT not null,
        l_quantity            DOUBLE PRECISION not null,
        l_extendedprice       DOUBLE PRECISION not null,
        l_discount            DOUBLE PRECISION not null,
        l_tax                 DOUBLE PRECISION not null,
        l_returnflag          CHAR(1) not null,
        l_linestatus          CHAR(1) not null,
        l_shipdate            DATE not null,
        l_commitdate          DATE not null,
        l_receiptdate         DATE not null,
        l_shipinstruct        CHAR(25) not null,
        l_shipmode            CHAR(10) not null,
        l_comment             VARCHAR(44) not null
);
```

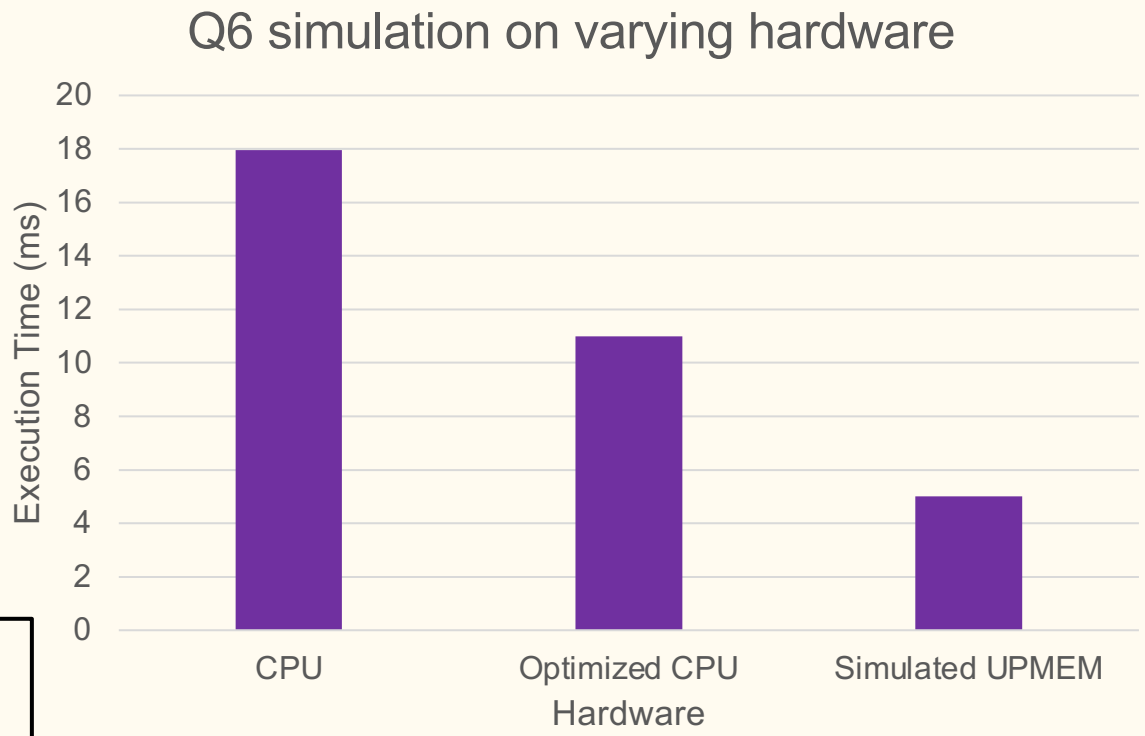
Naive CPU Q6 implementation in C

```
uint64_t tpch_q6(data* lineitem)
{
   uint64_t out = 0;
   for (size_t i = 0; i < NUM_TUPLES; i++) {
       if (   ((lineitem + j)->l_shipdate >= Q6_DATE1)
                && ((lineitem + j)->l_shipdate < Q6_DATE2)
                && ((lineitem + j)->l_discount >= Q6_DISCOUNT1)
                && ((lineitem + j)->l_discount <= Q6_DISCOUNT2)
                && ((lineitem + j)->l_quantity < Q6_QUANTITY)) {
            out += (lineitem + i)->l_extendedprice * (lineitem + i)->l_discount;
       }
   }
   return out;
}
```

Process for Q6 on UPMEM PIM

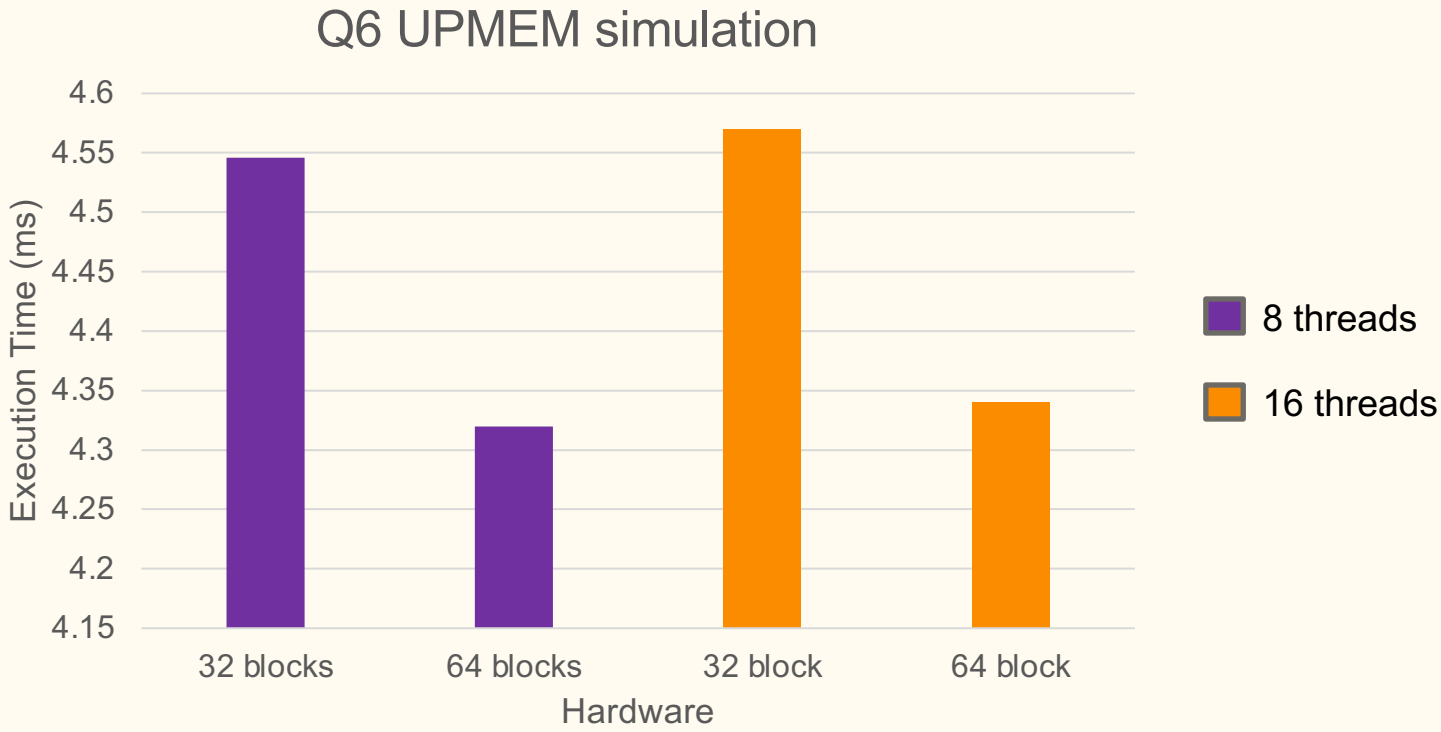
1. Load table into CPU DRAM with N tuples
2. Send `N / NR_DPUS` tuples (with filtering) to each DPU MRAM
3. Load `DPU_TUPS / BLOCKSIZE` into WRAM
    a. Only a total of 64 Kb shared among tasklets
    b. Loop until all `DPU_TUPS` are processed
4. Output is in WRAM local for each tasklet
5. Store local results from each tasklet into MRAM
6. Complete parallel reduction of received outputs utilizing the CPU for inter-DPU communication

6001215 Tuples on Q6



16 tasklets per DPU
32 tuple block size
32 DPUs

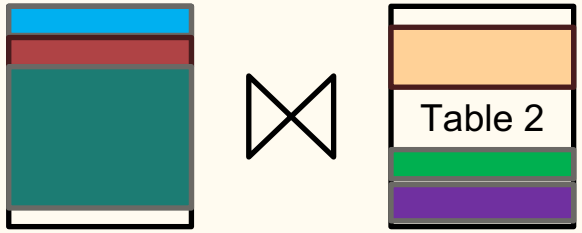
6001215 Tuples with 32 DPUs on Q6



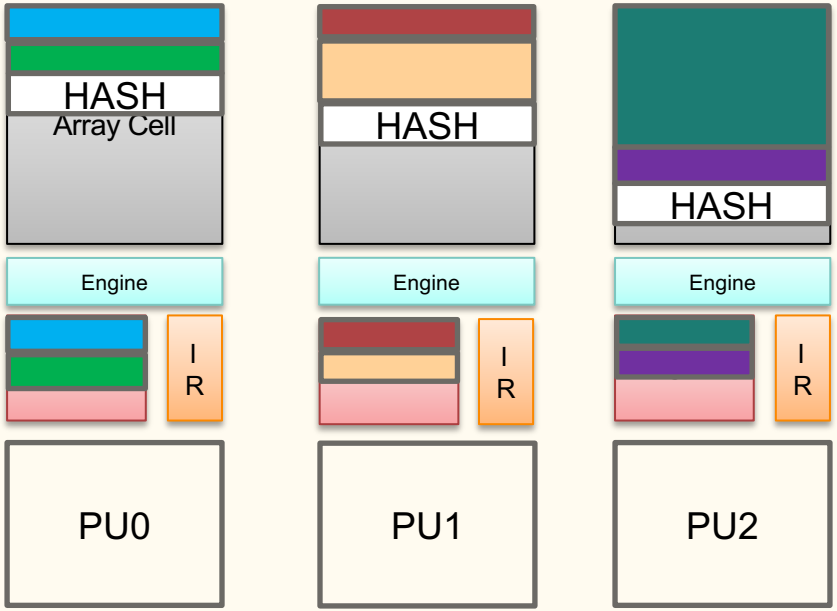
Q6 was **embarrassing** parallel

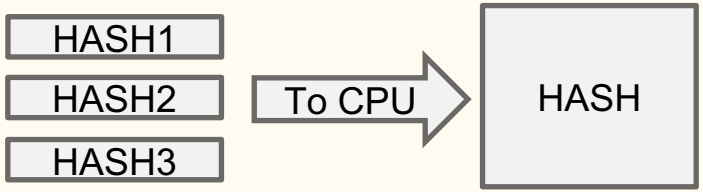
What if the queries required more communication?

How to do a join?



Utilize the **CPU** as a **co-processor** on UPMEM DIMMs for **high communication** tasks

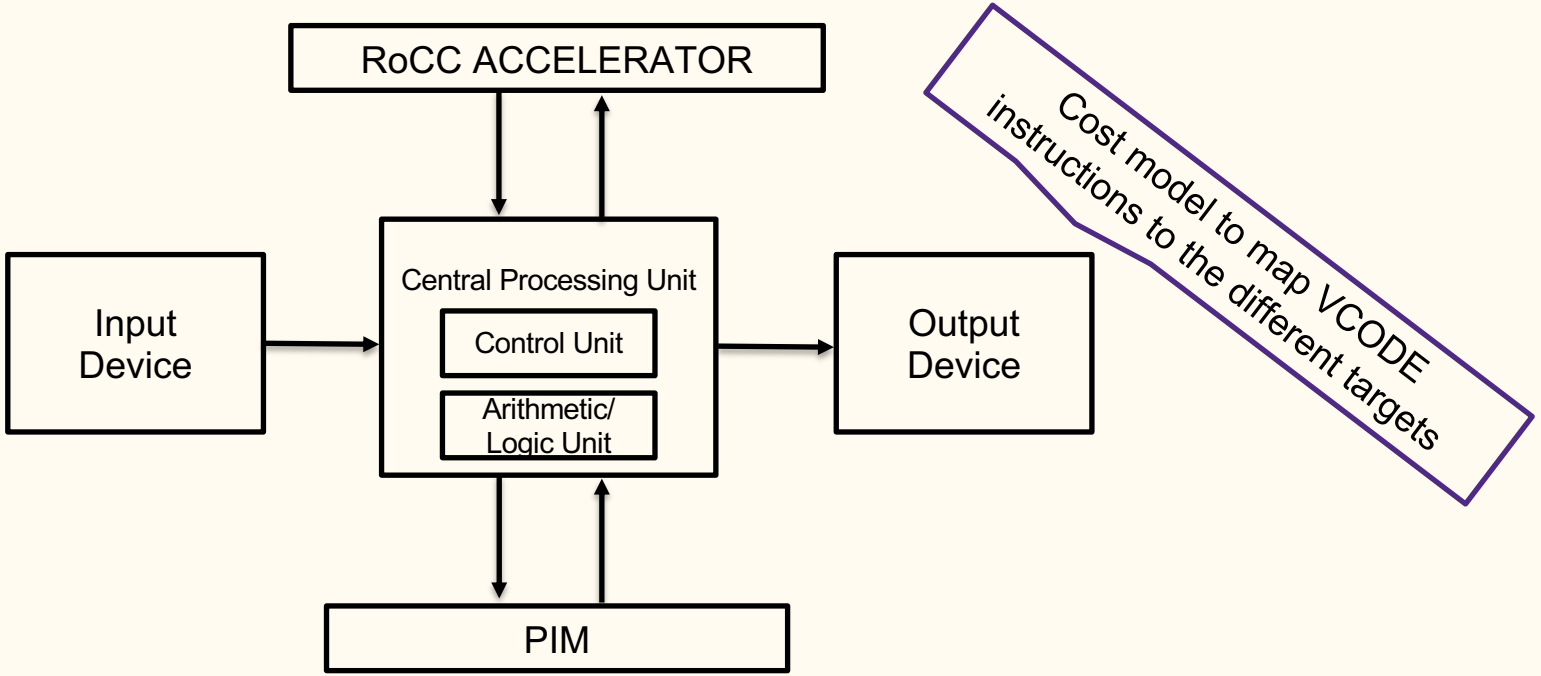




Village toolchain

- NESL, a high-level parallel language, has a parallel IR called VCODE
- Ability to compile VCODE to a variety of LLVM-supported architectures
- Added support for RISC-V RoCC accelerators

Heterogenous targets in Village



Conclusions and Future work

- UPMEM performs well with **low communication** and **high** data **parallelism**
- Communication problems can be mitigated by using **CPU** as a **co-processor**
- Add UPMEM codegen as a target for Village toolchain
- Cost model to map a high-level operator to a heterogenous target
- Look at other available PIM hardware

Questions?

krasow@u.northwestern.edu
krasow.dev