Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

# The RISC-V Vector processor in EPI

**Prof. Jesús Labarta**

BSC & UPC

IEEE Computer Society Israel Webinar

Barcelona, Jan. 11th 2022

# Age before beauty (disclaimer)

- Behavior (insight/models) before syntax
- Detail performance analytics before aggregated profiles
- Work instantiation and order before overhead
- Malleability before fitted rigid structure
- Possibilities before how tos
- Elegance before one day shine
- Power before force

- All about programmer mindset !!!

El abuelo cebolleta ataca de nuevo

# Outline

- A vision on HPC
- Towards Holstic Co-design
- EPI
  - Architecture
  - Software Development Vehicles (SDVs)
- Conclusion

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

# A vision on HPC

# The programming revolution

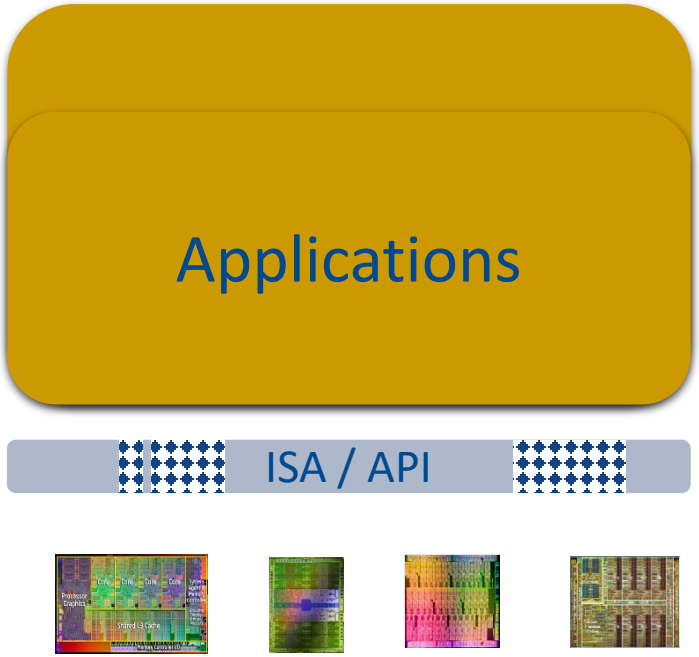
- **From the latency age …**
  - I need something … I need it now !!!
  - Performance dominated by latency in a broad sense
    - At all levels: sequential and parallel
    - Memory and communication, control flow, synchronizations

- **… to the throughput age**
  - Ability to instantiate “lots” of work and avoid stalling for specific requests
    - I need this and this and that … and as long as it keeps coming I am ok
    - At all levels
  - Performance dominated by overall availability/balance of resources

# Vision

- The multicore and memory revolution
  - ISA leak …
  - Plethora of architectures
    - Heterogeneity
    - Memory hierarchies

The power wall made us go multicore and the ISA interface to leak
→ our world is shaking

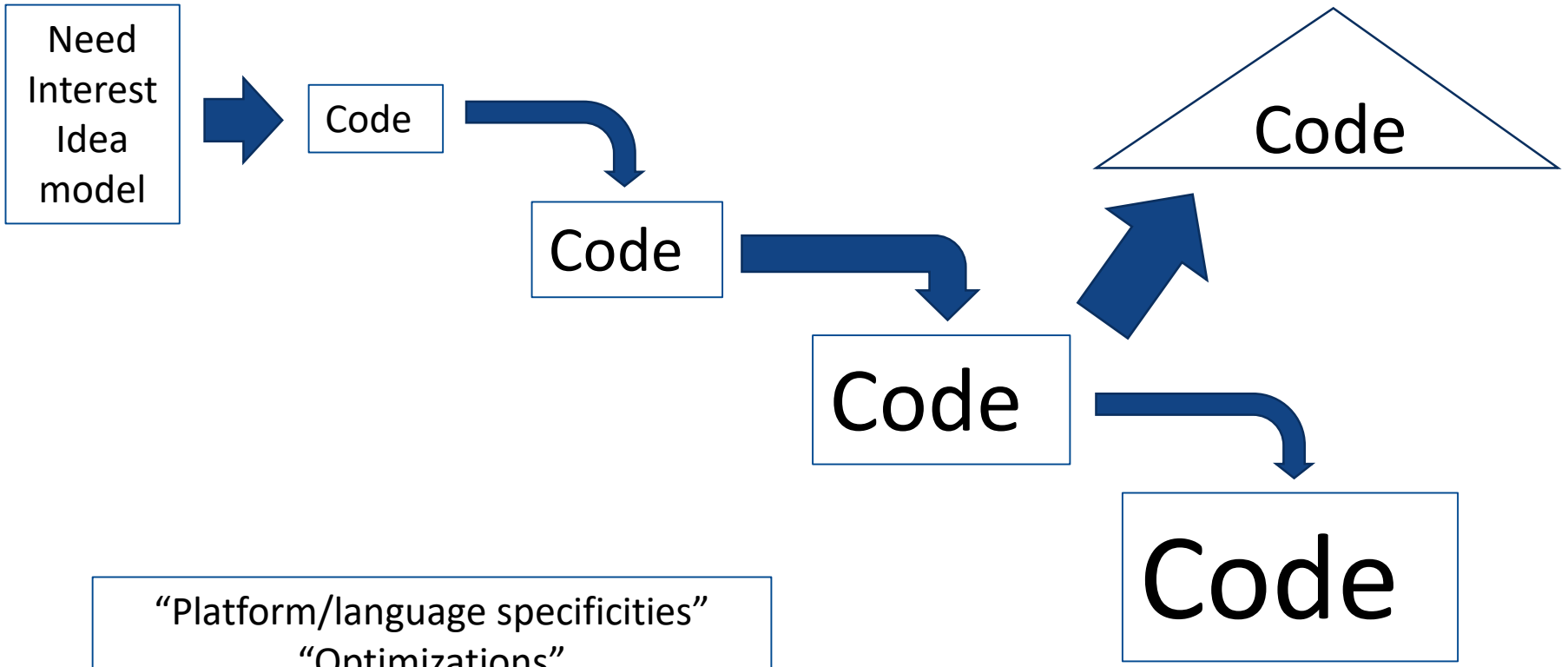



- Complexity + variability = Divergence
  - Between our mental models and actual system behavior

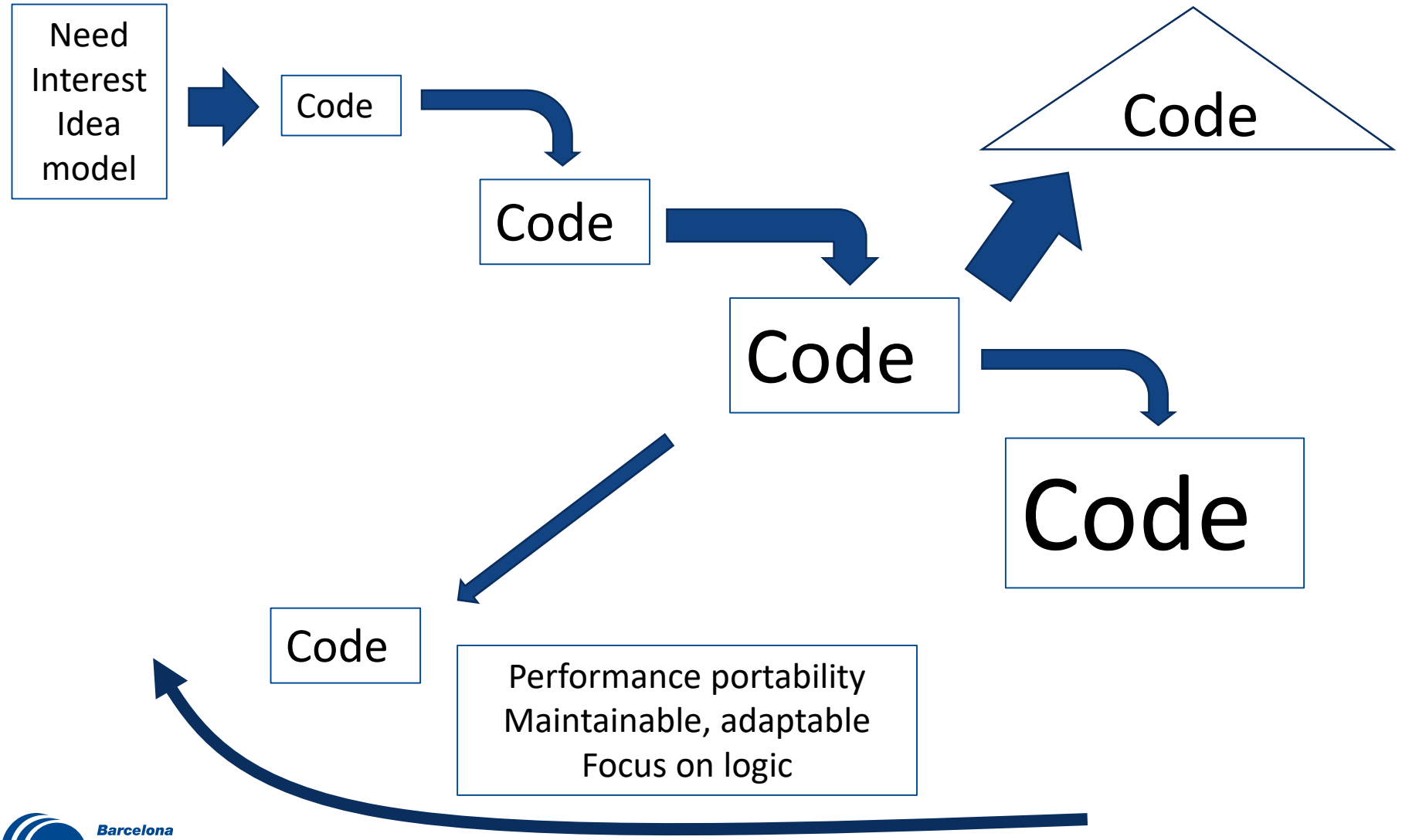
What programmers need ?

HOPE !!!

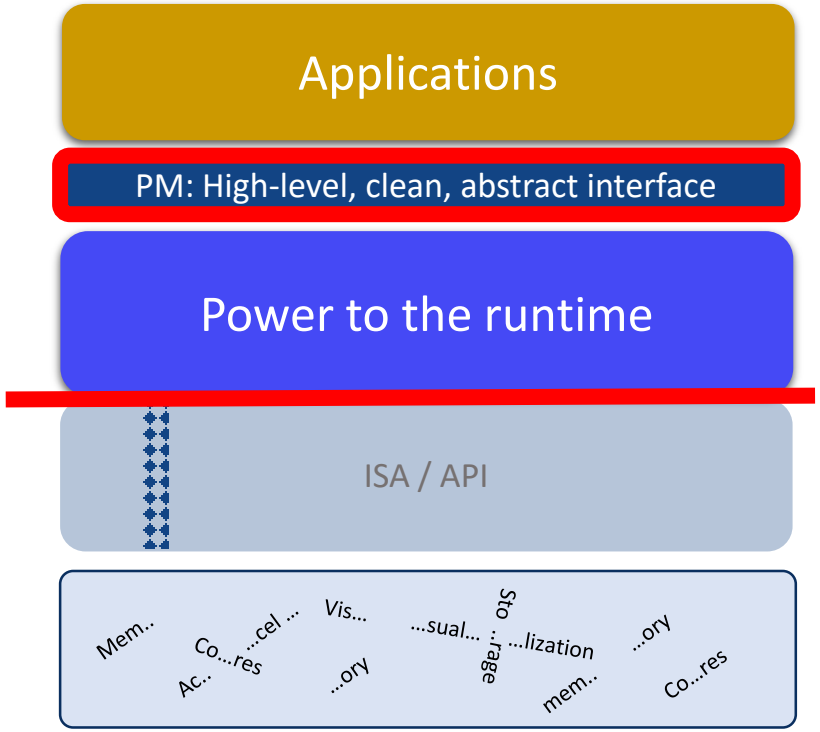
# Code lifetime cycle



# There is HOPE !!!



# The PM osmotic membrane




Vector ISA

Raise semantic level of interface

Possibility to reduce leakage

General purpose

Task & data based

Forget about resources

Decouple:
Minimal & sufficient permeability?

Intelligence
&
Resource management

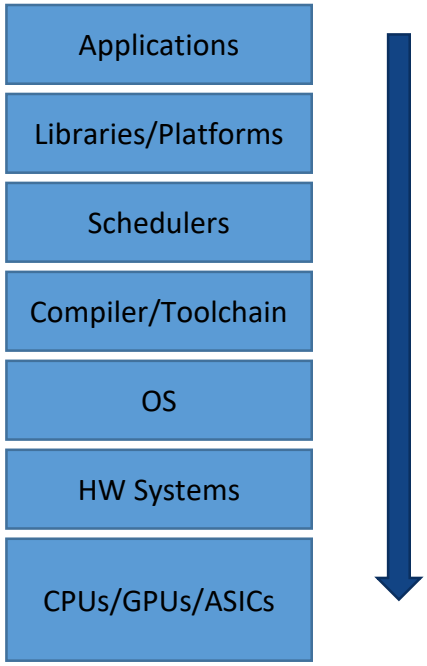
"Reuse & expand" old architectural ideas under new constraints

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

# Towards Holistic Co-design

# Co-design



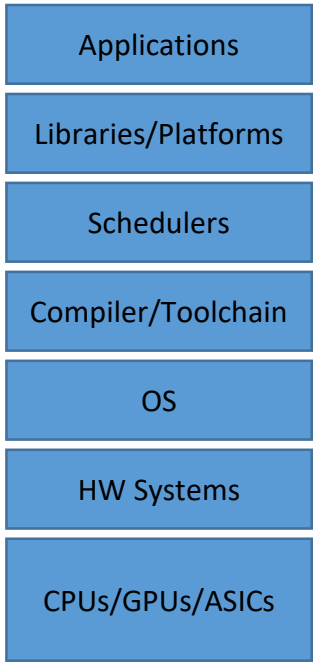

A buzzword !!!!

**NOT**: 1 user/customer + 1 vendor

NOT : Top → down !!!!
and obsessed by hardware

Limited scope !!!!

# Co-design




You have to DESIGN !!!!

Co-DESIGN vs. Co-DIMENSION !!!!

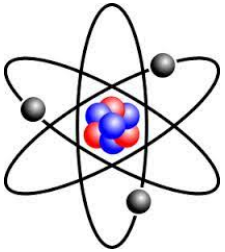
# Holistic co-design

- Applications
- Libraries/Platforms
- Schedulers
- Compiler/Toolchain
- OS
- HW Systems
- CPUs/GPUs/ASICs

"As above, so below"
Similar concepts/mechanisms at all levels

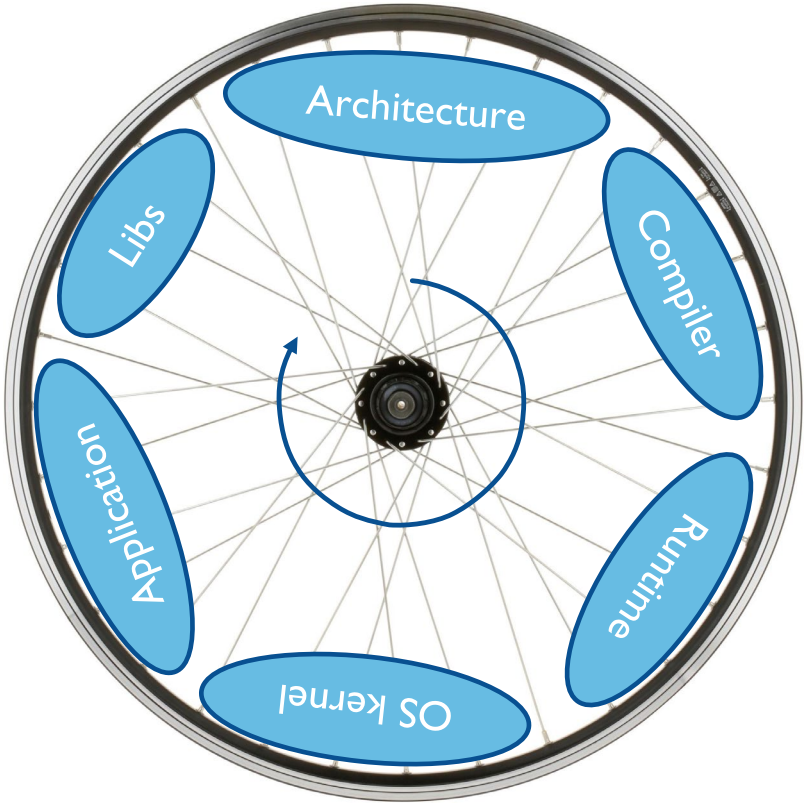
"Steered by a vision"
"Steered by detailed insight"

Principles
- Balanced hierarchy
- Latency → throughput: asynchrony & overlap
- Malleability and coordinated scheduling
- Homogenize heterogeneity



# Holistic Co-design

- Applications
- Libraries/Platforms
- Schedulers
- Compiler/Toolchain
- OS
- HW Systems
- CPUs/GPUs/ASICs




Best place to address an issue

Fundamentals

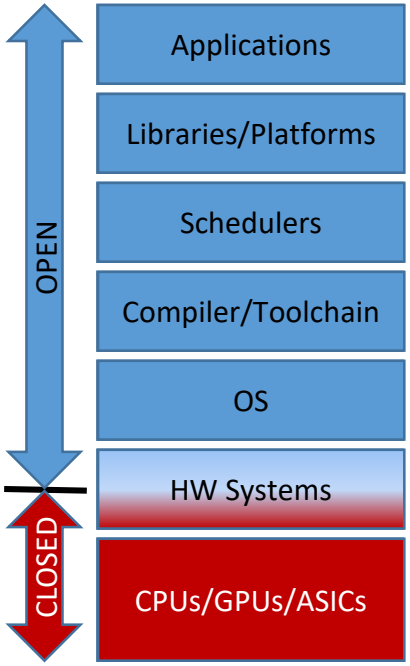
Balance

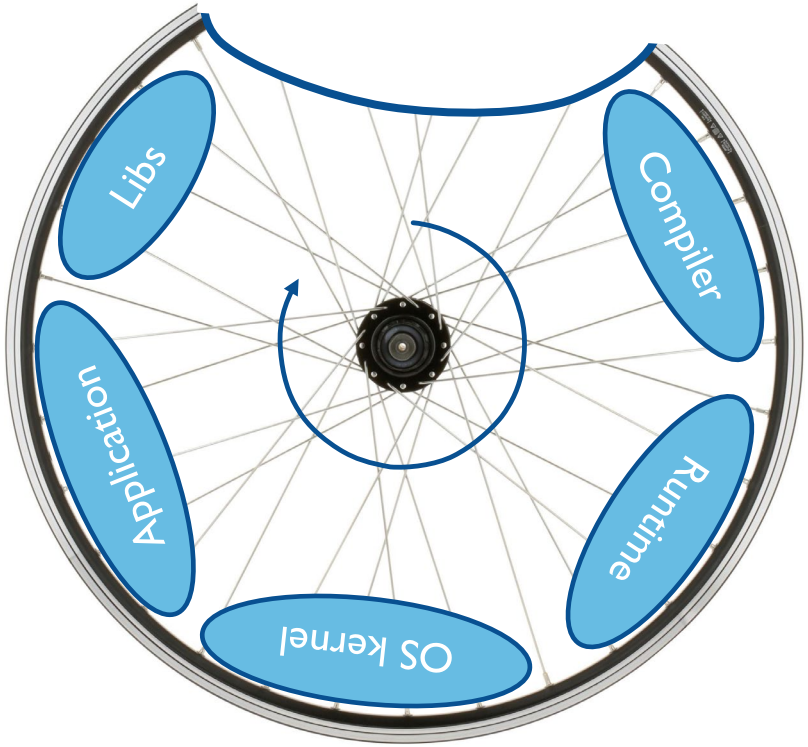
Mindset

Productivity

Efficiency

# Holistic Co-design




Best place to address an issue

Fundamentals

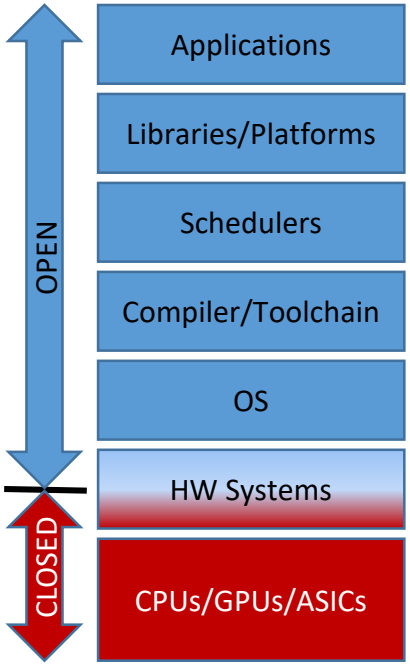
Balance

Mindset

Productivity

Efficiency

# Leverage interfaces and implementations



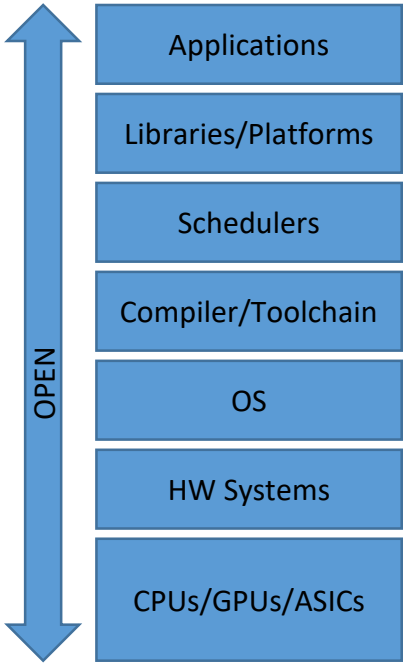

slurm

MPI

OpenMP

Leverage “standards”
Opportunity to innovate and contribute

# Leverage interfaces and implementations




slurm

MPI

OpenMP

RISC-V

Leverage “standards”
Opportunity to innovate and contribute

# Towads “Exascale”

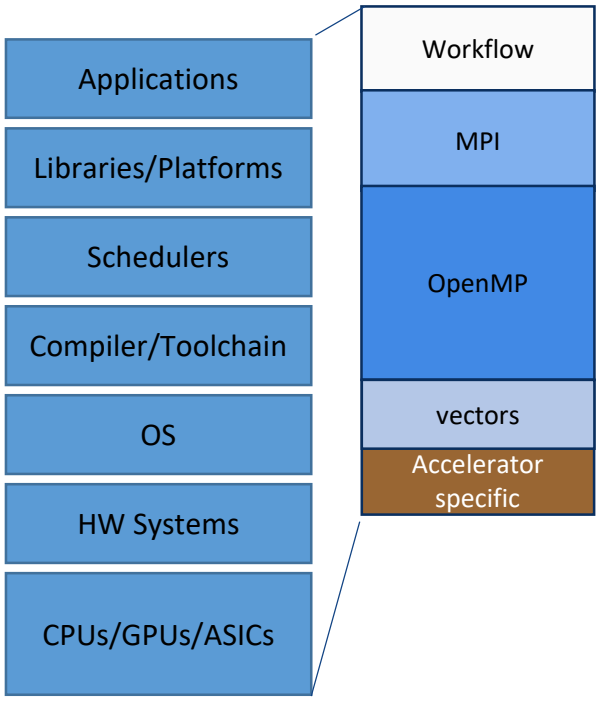
# principles 

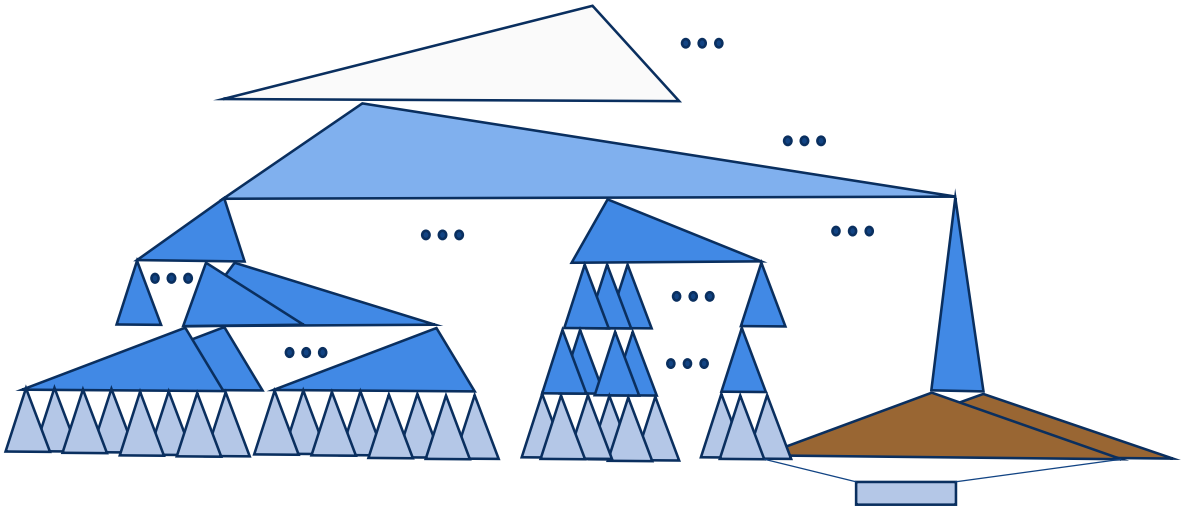
- Balanced hierarchy
- Throughput oriented: asynchrony and overlap
- Malleability and coordinated scheduling
- Homogenizing heterogeneity
- Detailed analysis and insight on behavior

# Balanced hierarchy

Expression & exploitation of Parallelism

$$10^6 = 1 \times 10^6 = 10 \times 10^5 = 10^2 \times 10^2 \times 10^2$$



Long vectors RISC-V

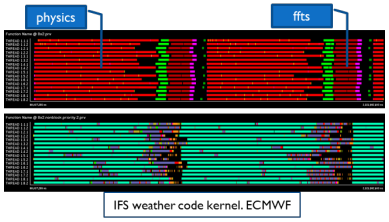
256 elements. 8 lanes per core
"Limited" number of "general purpose" control flows within tile

# Latency → Throughput: asynchrony and overlap

- Applications
- Libraries/Platforms
- Schedulers
- Compiler/Toolchain
- OS
- HW Systems
- CPUs/GPUs/ASICs

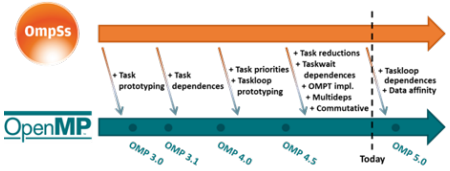
**Interoperability MPI + OpenMP**

- Taskify MPI calls

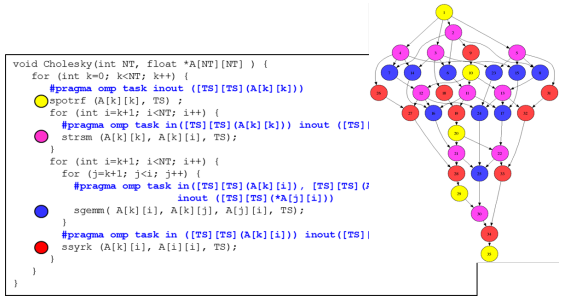



**Task based models**

- Single mechanism
  - Concurrency
  - Locality & data
  - ...gement




**Task based computational workflows**

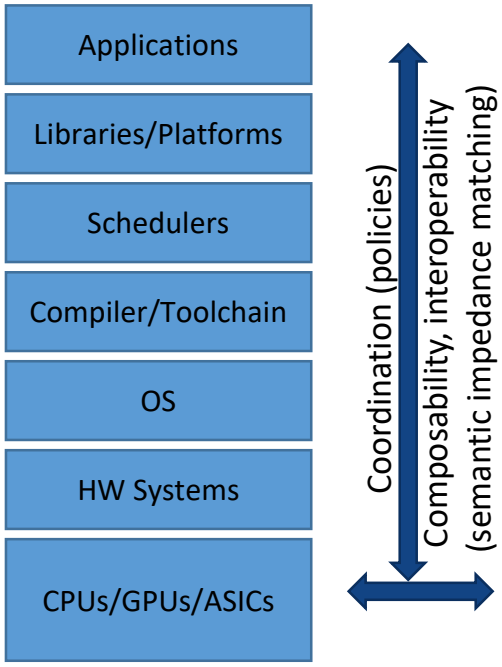



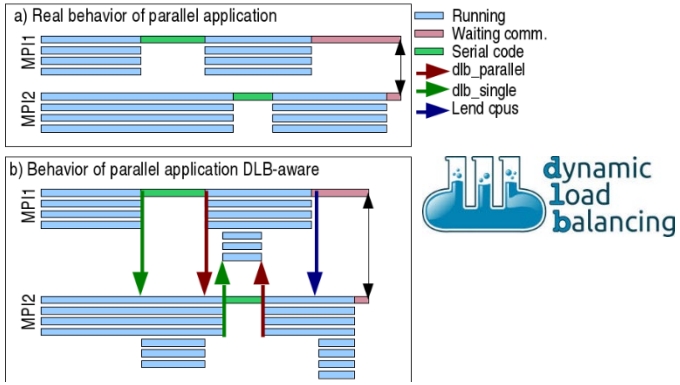
**Long vectors**

- decouple Front end – back end
- Convey access pattern semantics to the architecture. Potential to optimize memory throughput:

RISC-V

# Malleability & Coordinated scheduling




dynamic load balancing

A wish:
Handoff scheduling

**Vector Length Agnostic (VLA)**
programming and architecture

ISA

RISC-V

Micro architecture decides

```
saxpy:
    vsetvli a4, a0, e32, m8
    vlw.v v0, (a1)
    sub a0, a0, a4
    slli a4, a4, 2
    add a1, a1, a4
    vlw.v v8, (a2)
    vfmacc.vf v8, fa0, v0
    vsw.v v8, (a2)
    add a2, a2, a4
    bnez a0, saxpy
    ret
```

# Homogenizing Heterogeneity

- Applications
- Libraries/Platforms
- Schedulers
- Compiler/Toolchain
- OS
- HW Systems
- CPUs/GPUs/ASICs

```
void axpy_omp_nest    (double a, double *dx, double *dy, int n) {
   int i, chunk;
   #pragma omp taskloop
   for (i=0; i<n; i+=TS) {
      chunk= n>i+TS? TS : n-i;
      #pragma omp target map(to:dx[i:i+chunk], tofrom:dy[i:i+chunk])
      axpy_omp        (a, &dx[i], &dy[i], chunk);
   }
}
```

```
void axpy_omp         (double a, double *dx, double *dy, int n) {
   int I, chunk;
   #pragma omp taskloop
   for (i=0; i<n; i+=TS) {
      chunk= n>i+TS? TS : n-i;
      axpy_SIMD       (a, &dx[i], &dy[i], chunk);
   }
}
```

```
void axpy_SIMD        (double a, double *dx, double *dy, int n) {
   int i;
   #pragma omp simd
   for (i=0; i<n; i++) dy[i] += a*dx[i];
}
```

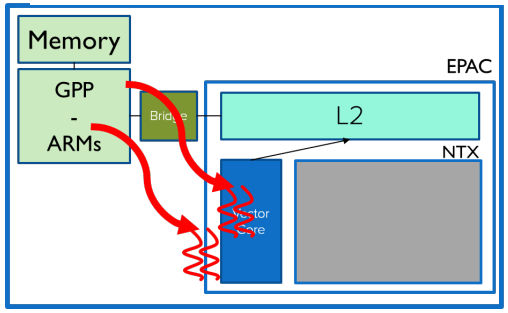
**VLA helps homogenize Heterogeneous Performance**

- ~ Big – Little cores, ….

**Nested tasked/workshared**

OpenMP

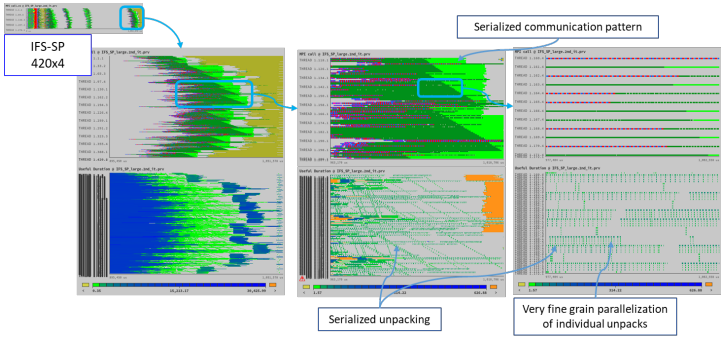
- Offload regular OpenMP

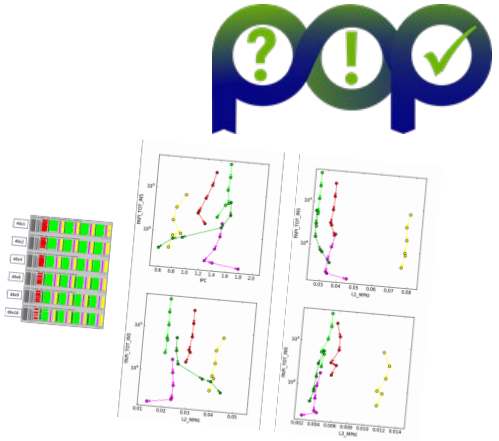


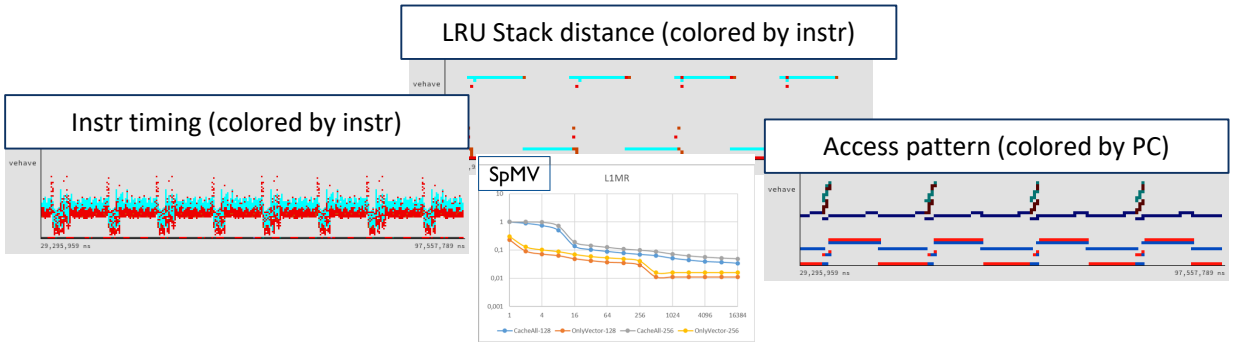

- HW support: IO coherence

# Detailed analysis and Insight on behavior

- Applications
- Libraries/Platforms
- Schedulers
- Compiler/Toolchain
- OS
- HW Systems
- CPUs/GPUs/ASICs



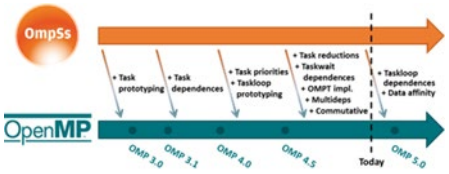





# Some co-design related activities @ BSC

- Performance tools & analysis methodology
  - POP

- Programming model
  - OmpSs & OpenMP




- RISC-V Vector Architecture
  - EPI

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

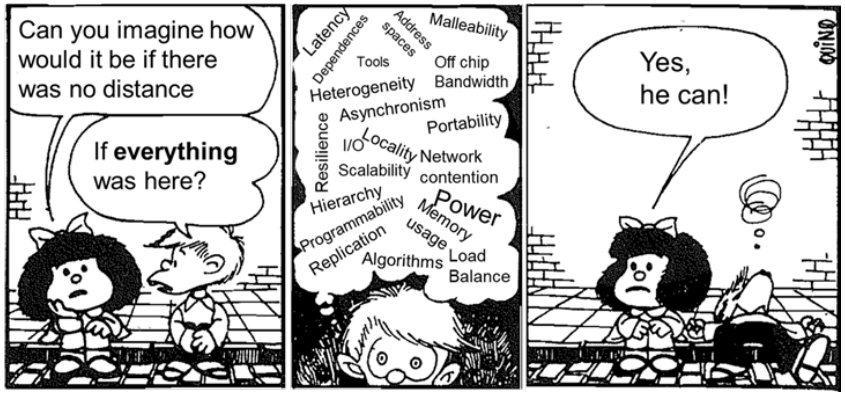
# EPI

# The EPI FPA Objective

- Components (low power microprocessor technologies) …
  - ARM based SoC
  - RISCV based accelerator

- … to be combined to target
  - HPC
  - HPDA
  - Emerging
    - Automotive
    - …

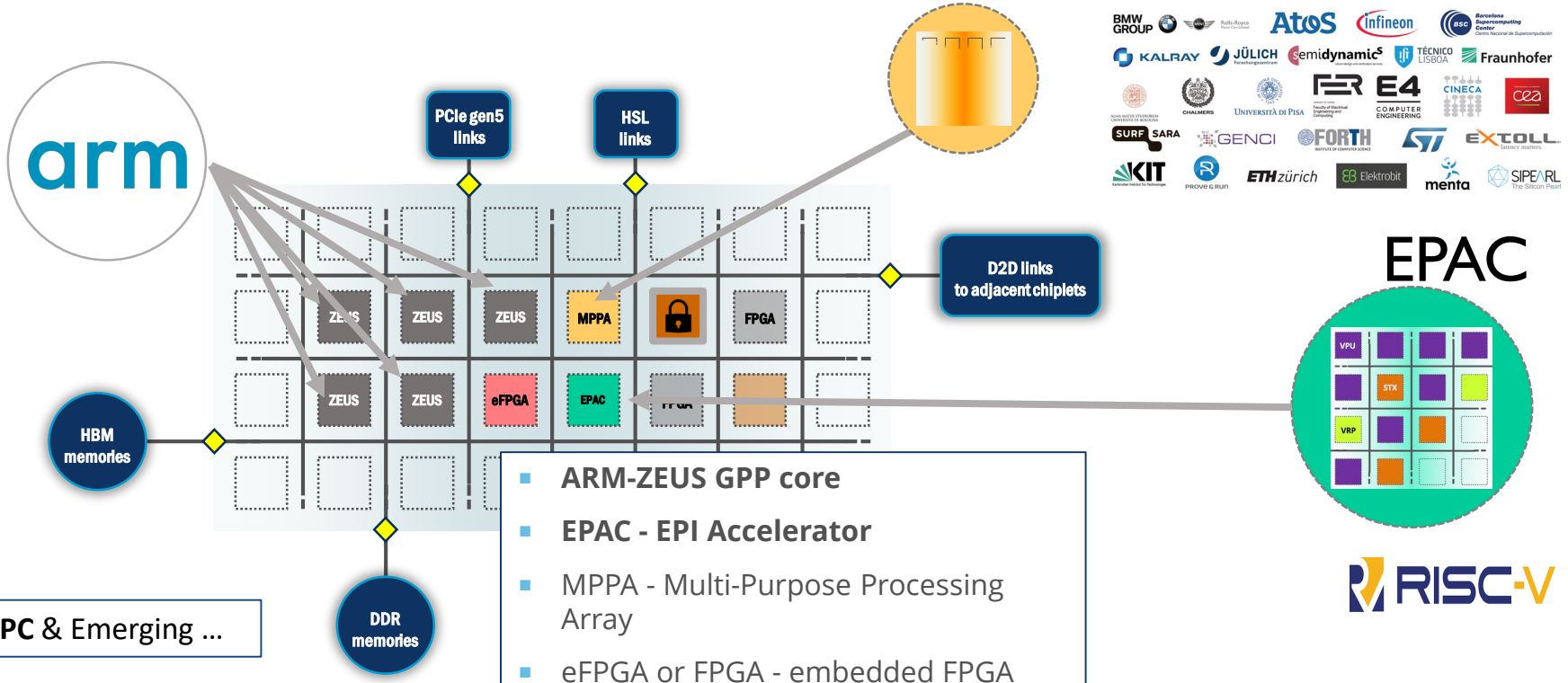
European Processor Initiative



# Three streams

- General purpose
  - ARM SVE
  - BULL: System integrator → chip integrator → SiPearl
- Accelerator
  - RISC-V
  - EU design: BSC, Semidynamics, EXTOLL, ETHZ, UNIBO, Chalmers, …
- Automotive
  - Infineon, …

# Overall architecture

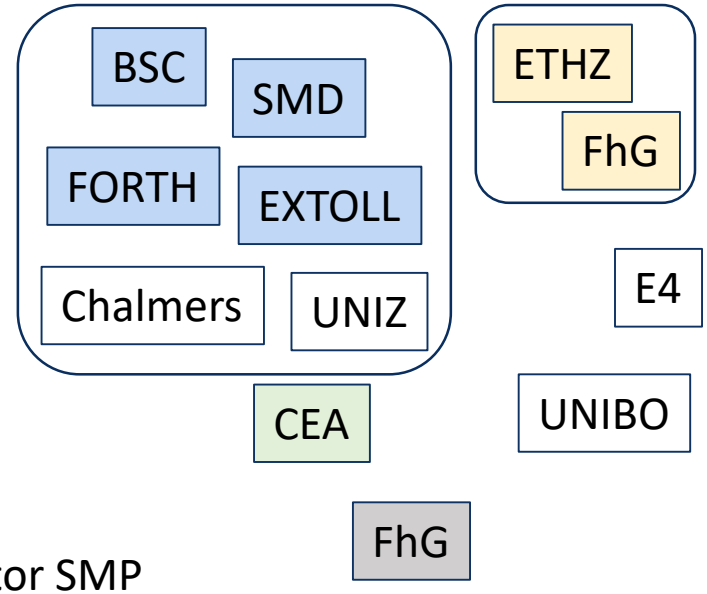



- **ARM-ZEUS GPP core**
- **EPAC - EPI Accelerator**
- MPPA - Multi-Purpose Processing Array
- eFPGA or FPGA - embedded FPGA
- Cryptographic ASIC (EU Sovereignty)

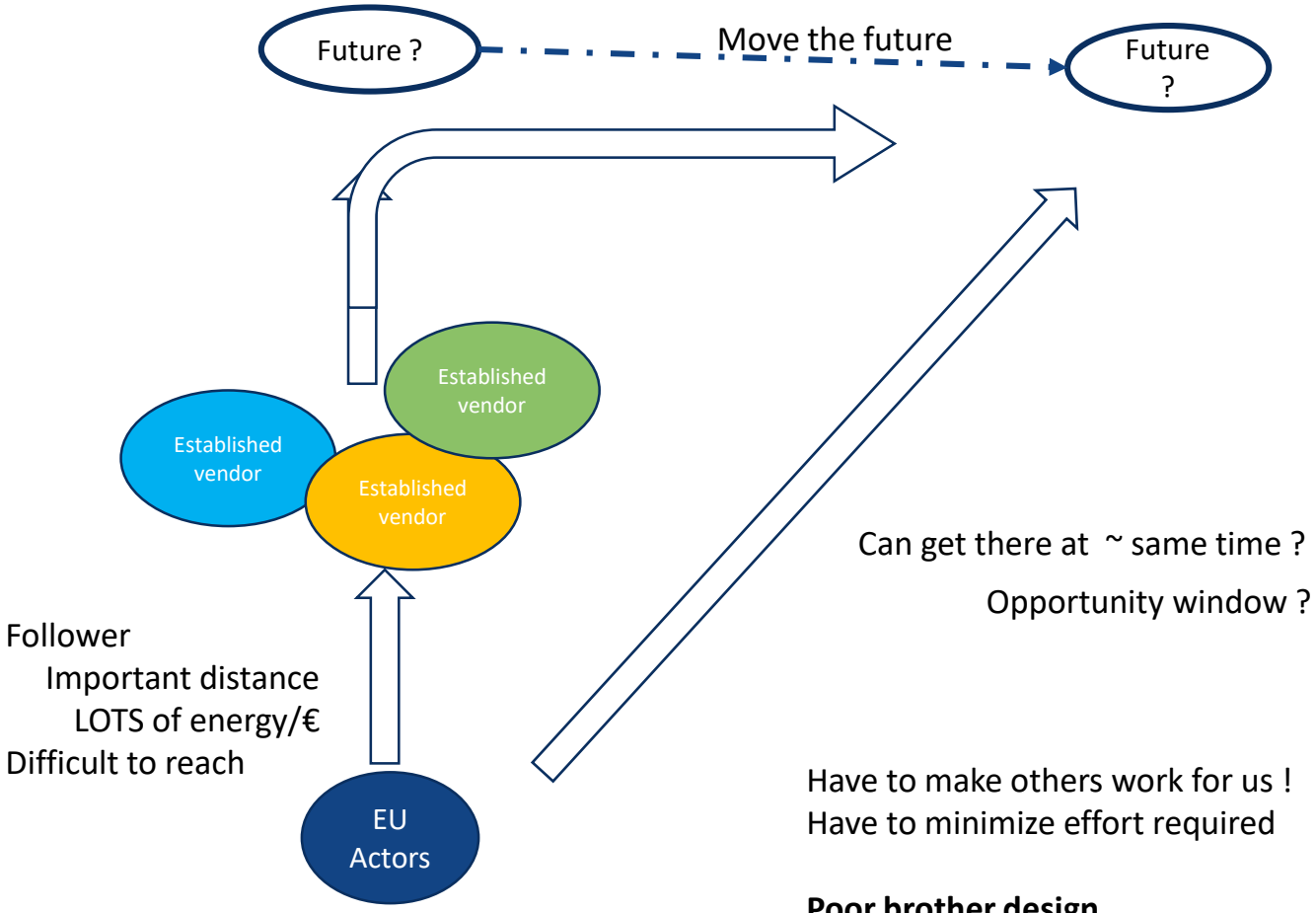
**HPC** & Emerging …

# Visions and collaborations

- STX:
  - Specific Accelerator devices
    - AI
    - Stencil
- RVV
  - ISA is important, RISC-V Vector
  - “Accelerator”
    - Easier entry, focus
    - → Standard self hosted, general purpose vector SMP
- VRP:
  - Extended precision arithmetic

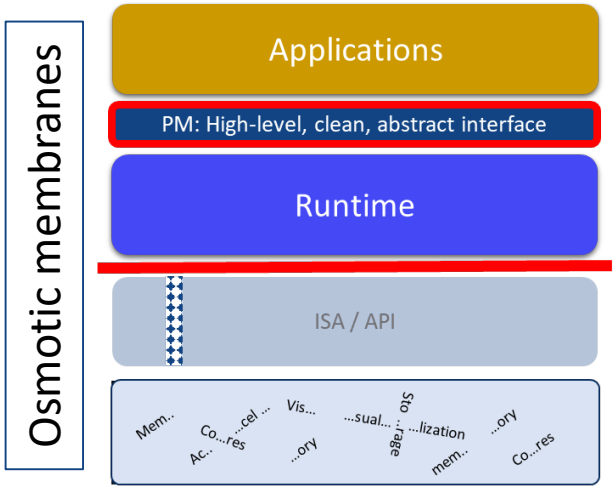


# The importance of a vision

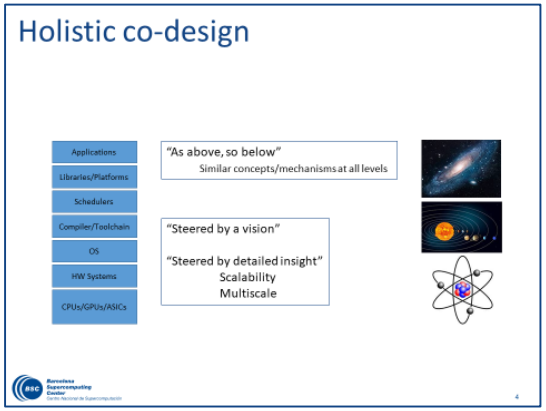


# RISCV accelerator vision @ EPI

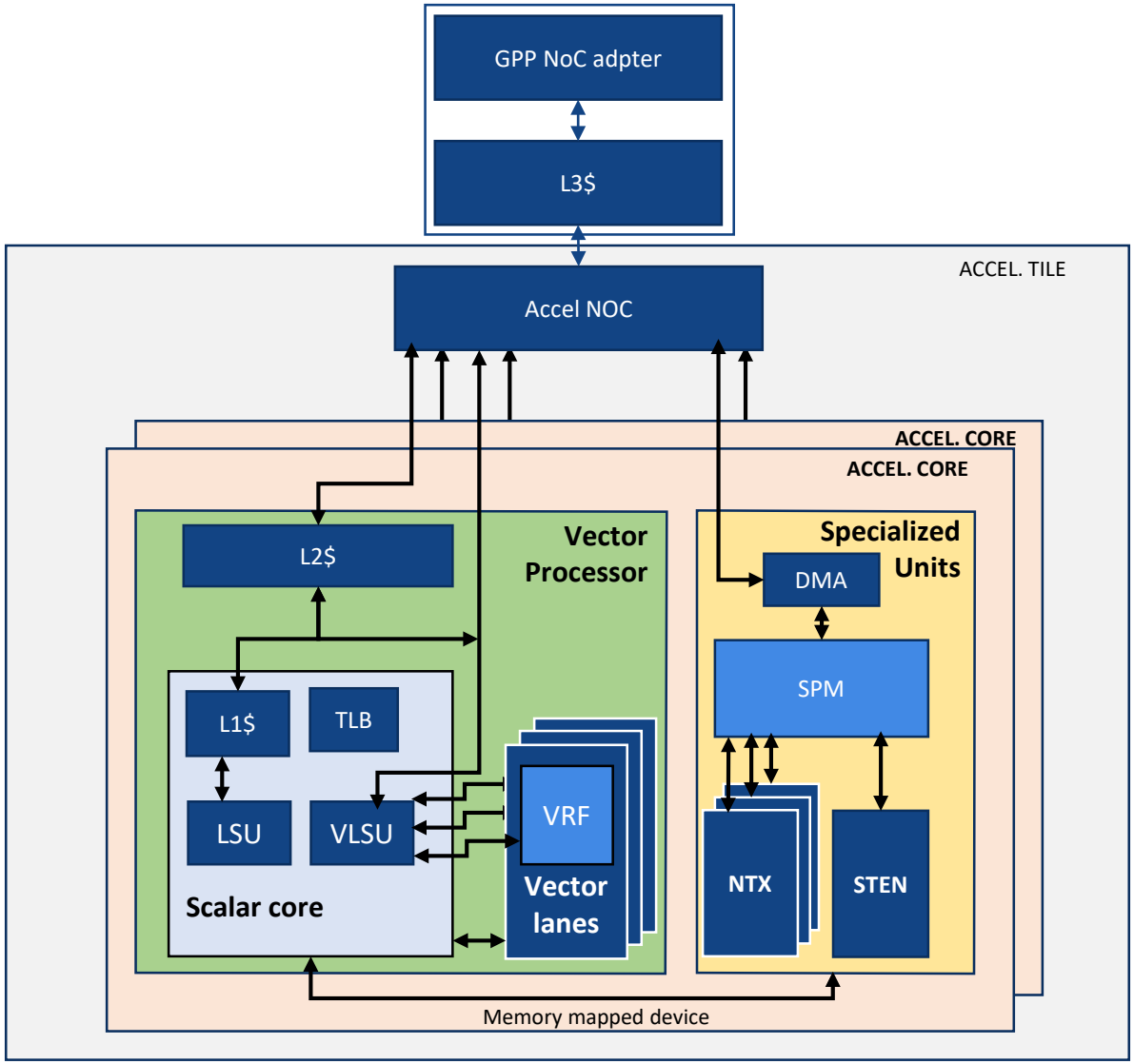

- **High throughput devices (processors)**
  - **Long Vectors**
    - "less words, more work"
    - Optimize memory throughput (High BW, B/F), locality
    - Decouple FE-BE
    - Locality & bandwidth at register level
  - **ISA is important**
    - Standard
    - name spaces: single linear coherent memory, registers
    - Vector Length Agnostic (VLA)
- **Hierarchical Acceleration**
  - Nesting
  - Balanced hierarchy: number of levels, ratio
    - **"limited" number of control flows**
  - Homogenized heterogeneity
- **Low power**:
  - ~ low voltage x ~ low frequency
  - **Efficiency**
- **MPI+OpenMP**
  - Throughput oriented programming approach
  - Malleability in application + Dynamic resource (cores, power, BW) management
  - Intelligent runtimes & Runtime Aware Architectures
    - Handle overlap and locality (improve B/F)
    - Reduce overhead → hierarchical
    - Architectural support for the runtime




Enabled by RISC-V

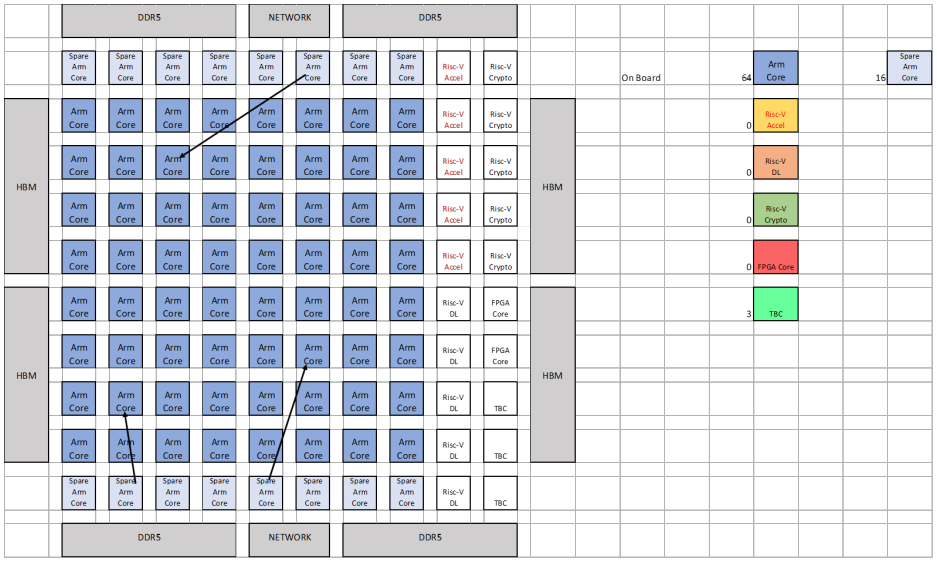


# “Original EPAC Architecture”

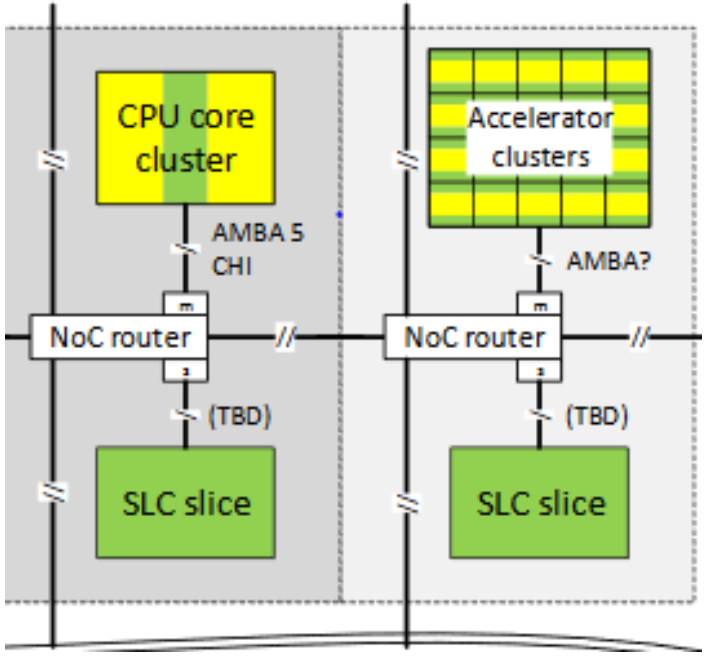


# GPP & Accelerator

- Integrate into Common Architecture



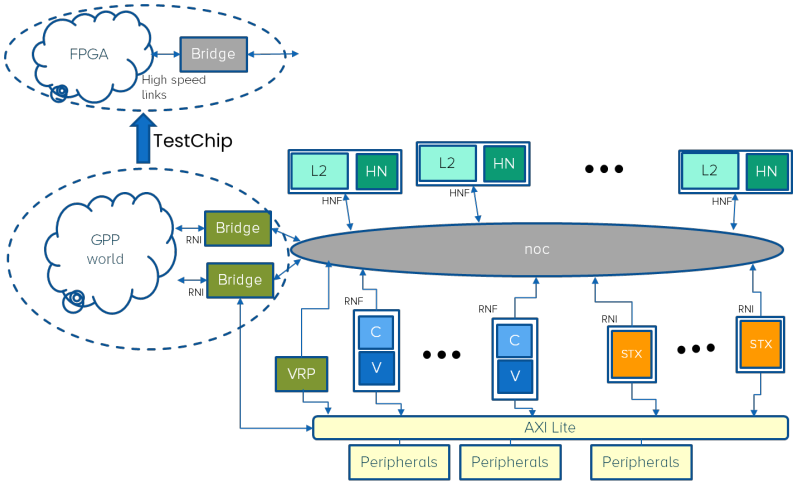

From Global Architecture/GPP Meetings



# EPAC architecture

RVV

- RV64GCV (→ 8x)
  - 2 way in order core
  - Decoupled VPU
    - 8 lanes
    - Long vectors (256 DP elements)
  - → 128 MSHR
  - L1 - MESI coherency
    - No allocate but coherent for vector L/S
- CHI interface NoC
  - 1 line / cycle
- L2$: 256KB/module
  - Allocation control mechanisms
- No in tile L3$




STX

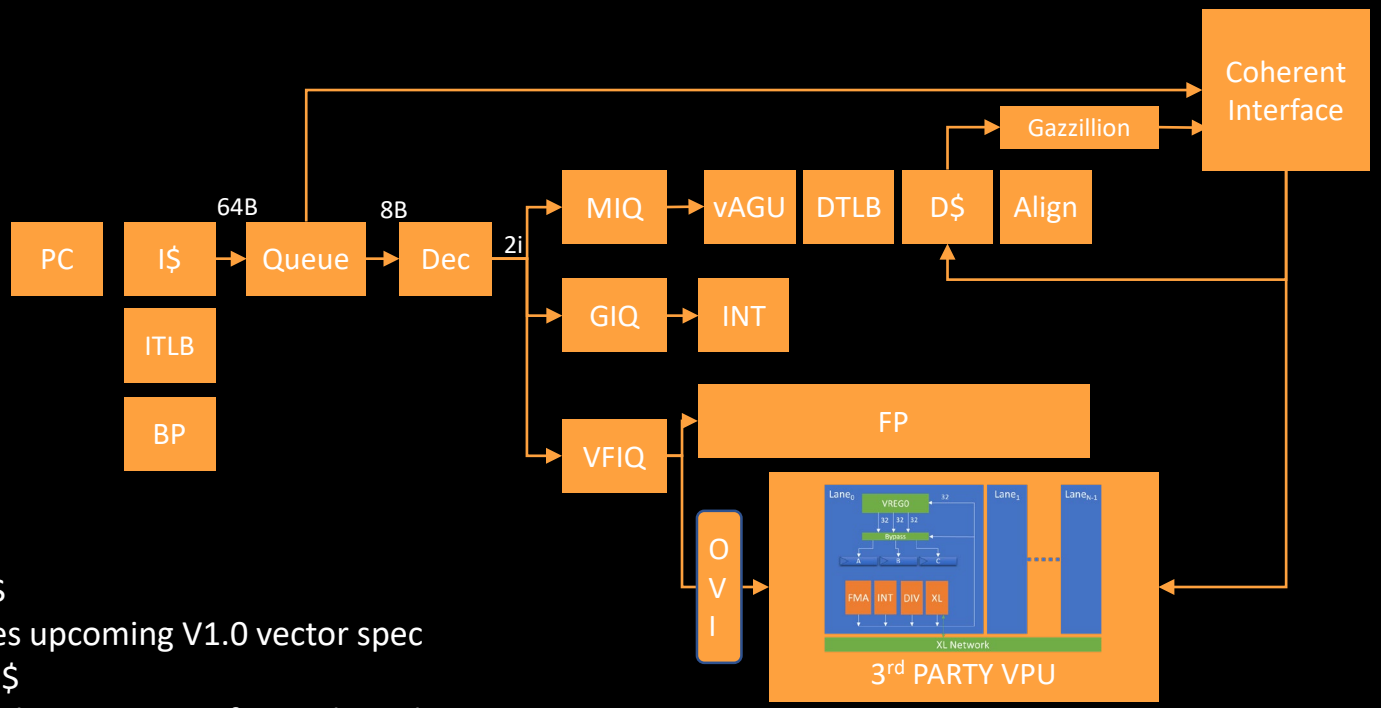
- DL and stencil specific accelerators
- Extensions to planned NTX
  - Programmable address generators →
  - lightweight RISC-v core + fat FPU + (Streaming Semantics & FR

VRP

- Variable precision processors

# AVISPADO 220 with VPU

RISCV64GCV

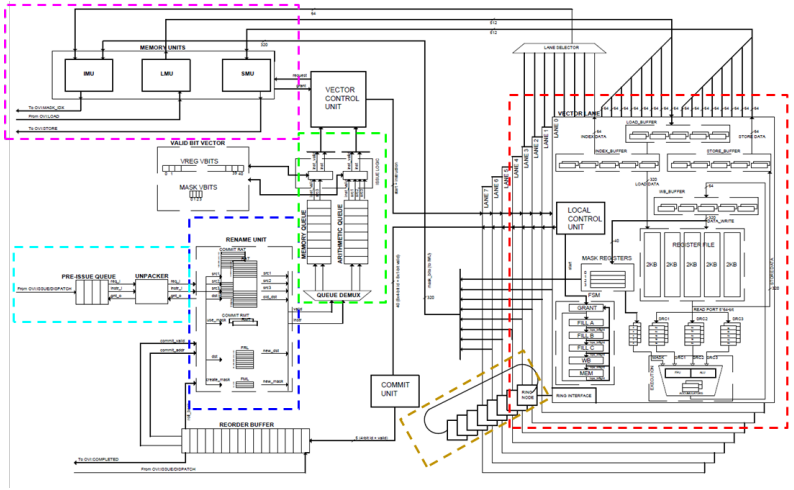



- SV48
- 16KB I$
- Decodes upcoming V1.0 vector spec
- 32KB D$
- Full hardware support for unaligned accesses
- Coherent (CHI)
- Vector Memory (vle, vlse, vlxe, vse, …) processed by MIQ/LSU

Courtesy R. Espasa

# VPU: A processor in itself

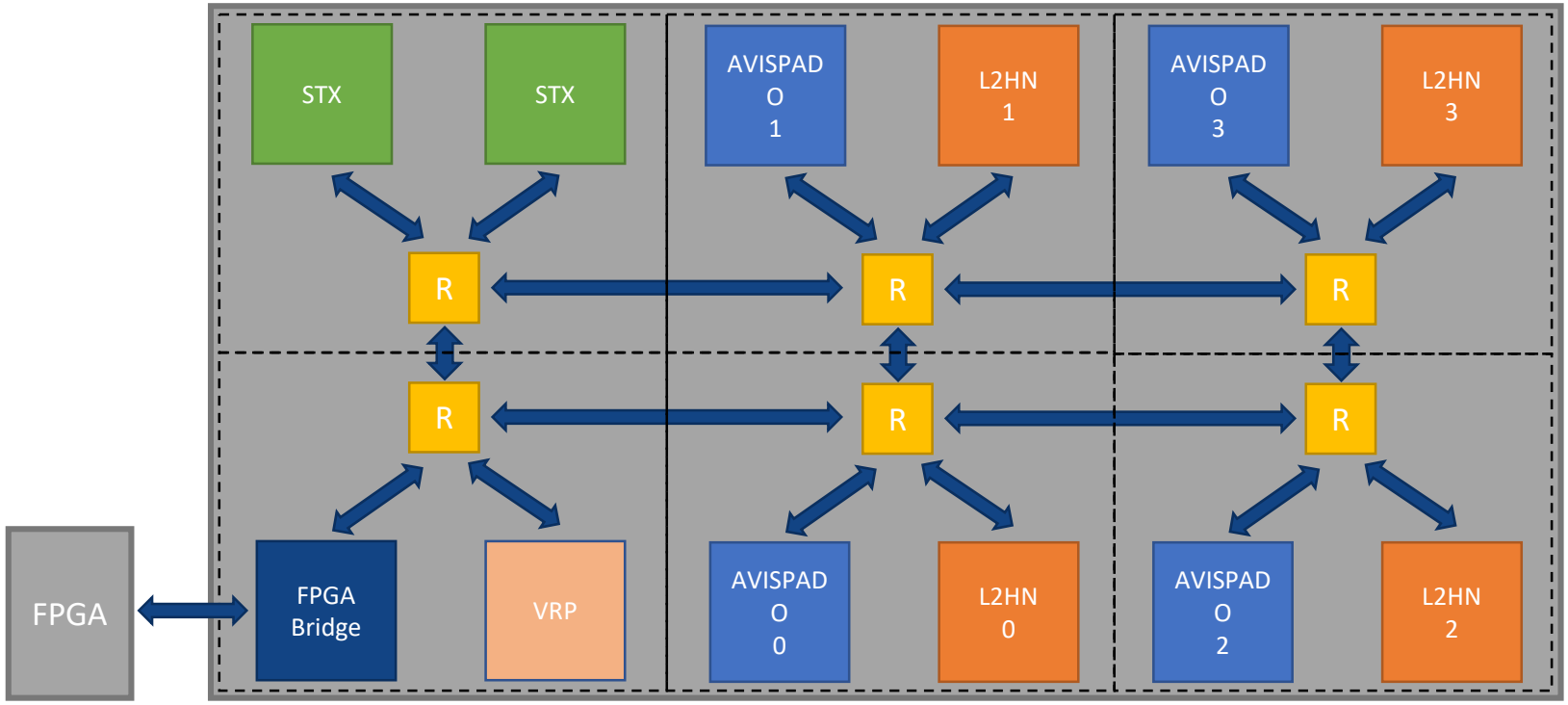
- Hierarchical “accelerator” integration
  - Program & data served by scalar core (Coherence; ~punch tape program ☺)
  - Fine grain “offloading” of “vector tasks” (directly hardware supported)
  - Homogenized heterogeneity under single “standard” ISA interface defining program order




- Implementation
  - **#FUs << VL** (lanes=8, VL=256)
  - Some OoO
    - Resources to overlap?
      - L/S, FU, shuffling
    - Renaming
      - 40 physical registers
  - Single ported register file
    - Large state
    - 5 banks/lane providing sufficient bandwidth for 1 op/cycle (latency/BW trading)
  - Data shuffling: directional ring

“Vitruvius: An Area-Efficient Decoupled Vector Accelerator for High Performance Computing”
F. Minervini, O. Palomar. RISC-V Summit 2021

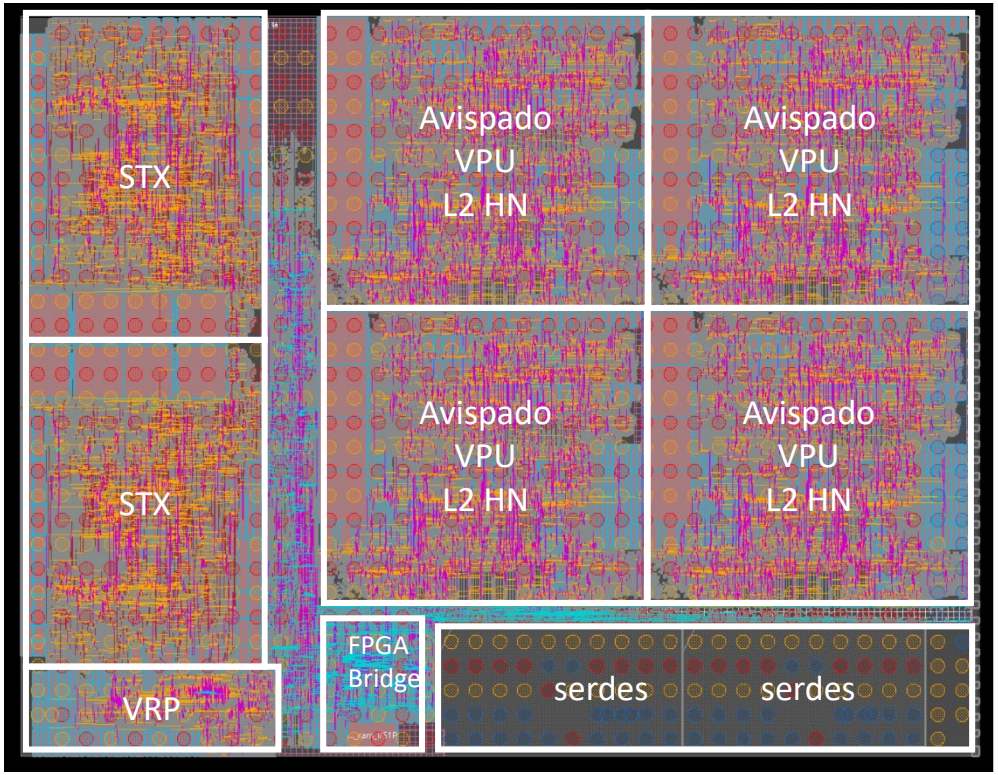
# EPAC Test Chip




Courtesy V. Papaefstathiou

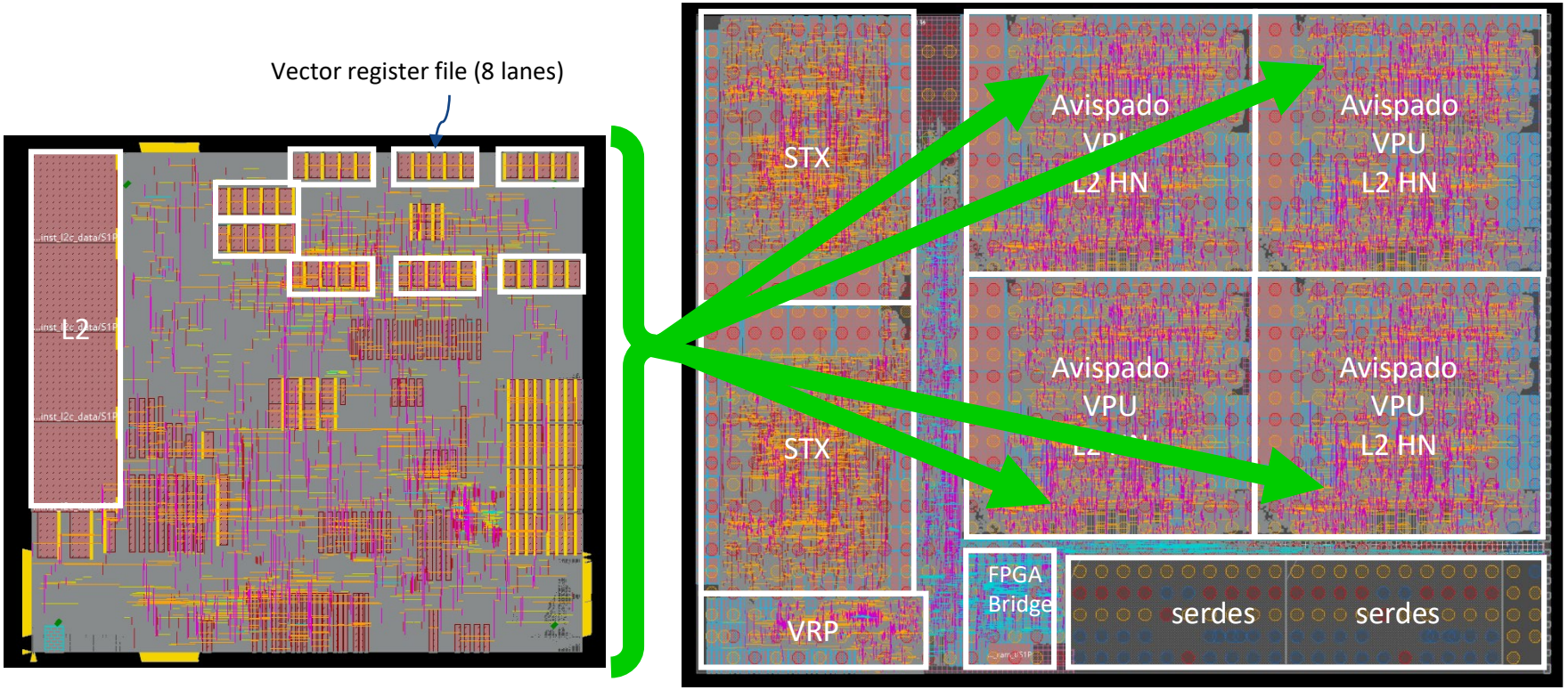
# Physical design

- GF22
- Final Top level chip floorplan
- Total area:
  - 5943 X 4593 um2
  - (27.297 mm2)



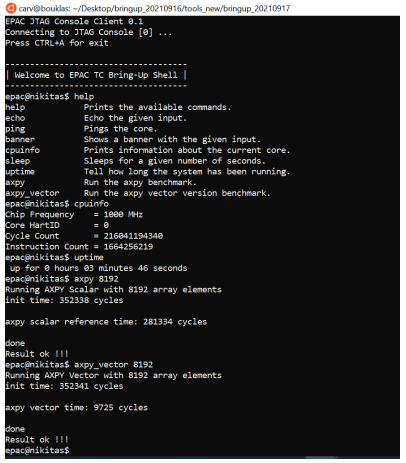
# Physical design

- 4 “VPU microtiles”
  - 2.517 mm2 each



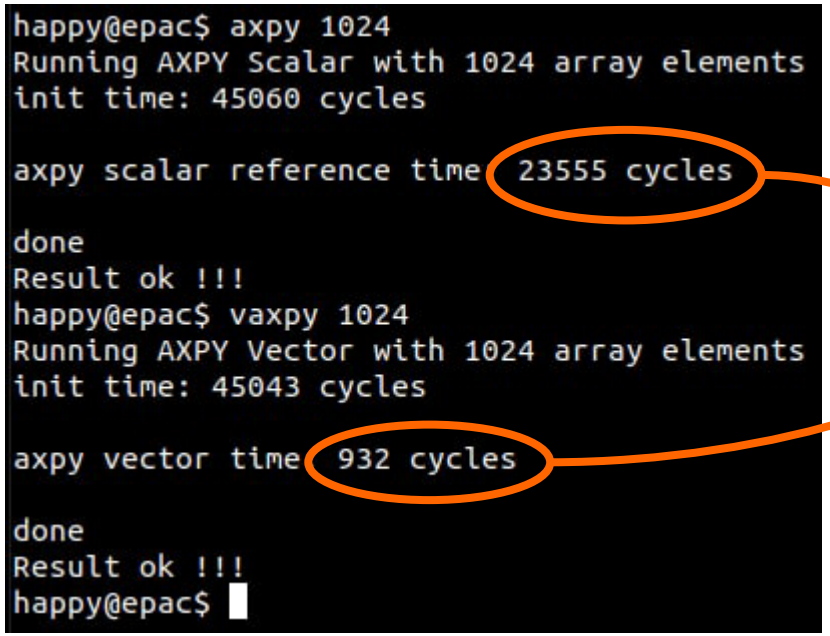
# Alive …

- Starting bring up process (sept 2021)
  - First words, …
  - … tiny shell …
  - … first vector code @ 1 GHz …


Special thanks to FORTH & EXTOLL

# Alive …




~25x
while only 8x FPUs
→ Long vectors !!

☺

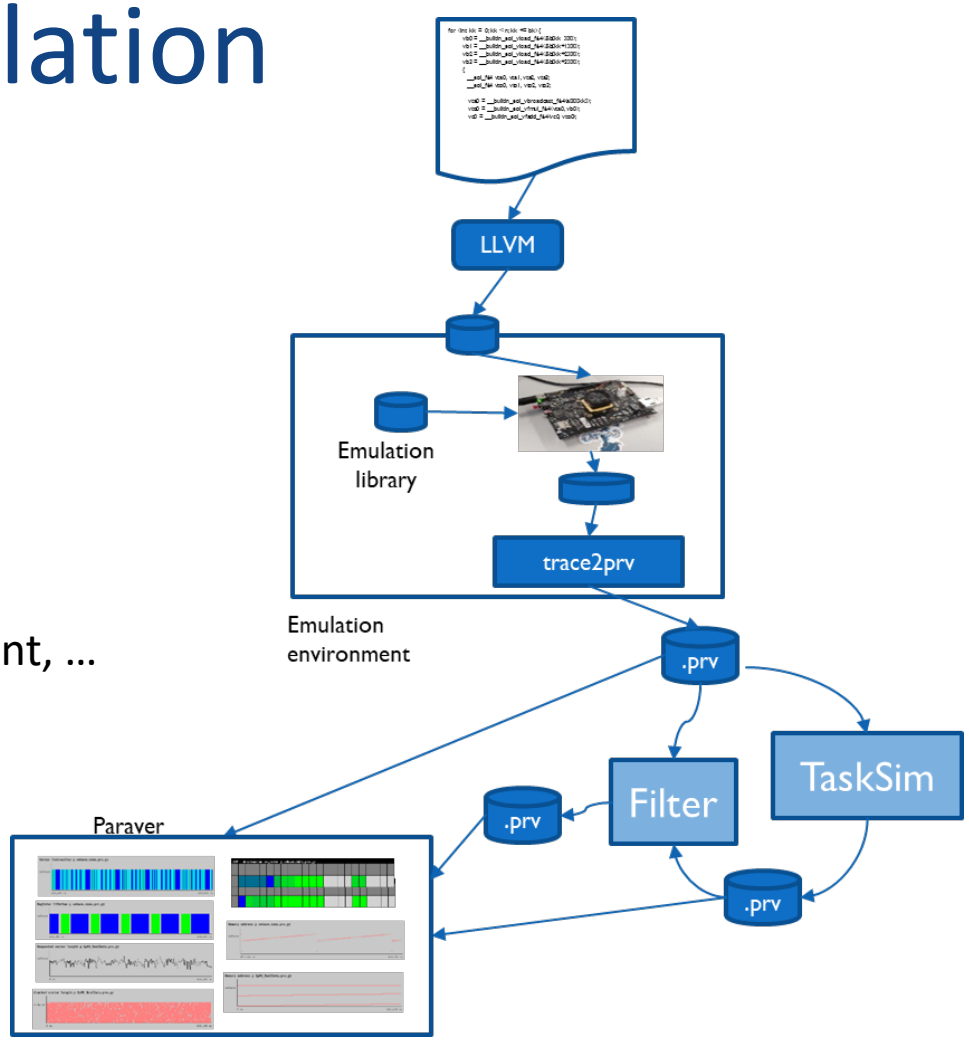
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

# SDVs

# RVV Software emulation

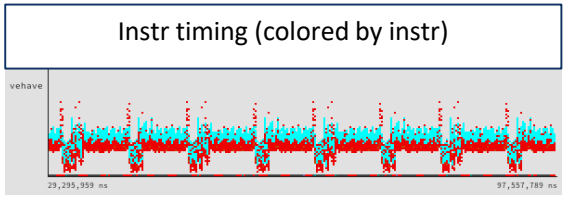
- Functional
  - Full OS, ….
  - On QEMU & scalar RISC-V cores
- Timing model
  - Microarchitectural parameters
    - MAXVL, OoO, Locality management, …

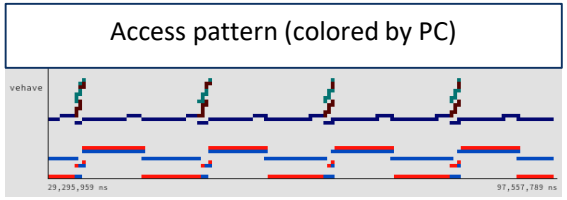


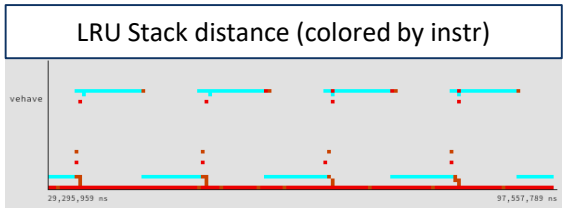

- https://repo.hca.bsc.es/gitlab/epi-public/risc-v-vector-simulation-environment

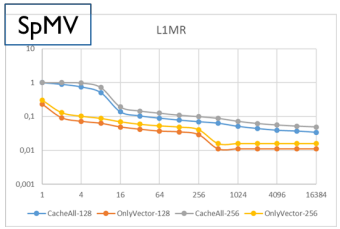
# RVV Software emulation

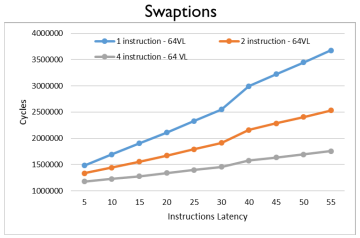
- Detailed analysis & insight



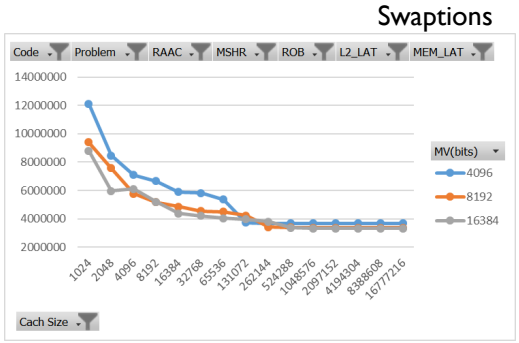



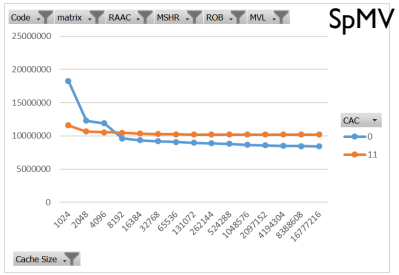




32KB



# RVV Software emulation

- Yolo: AI

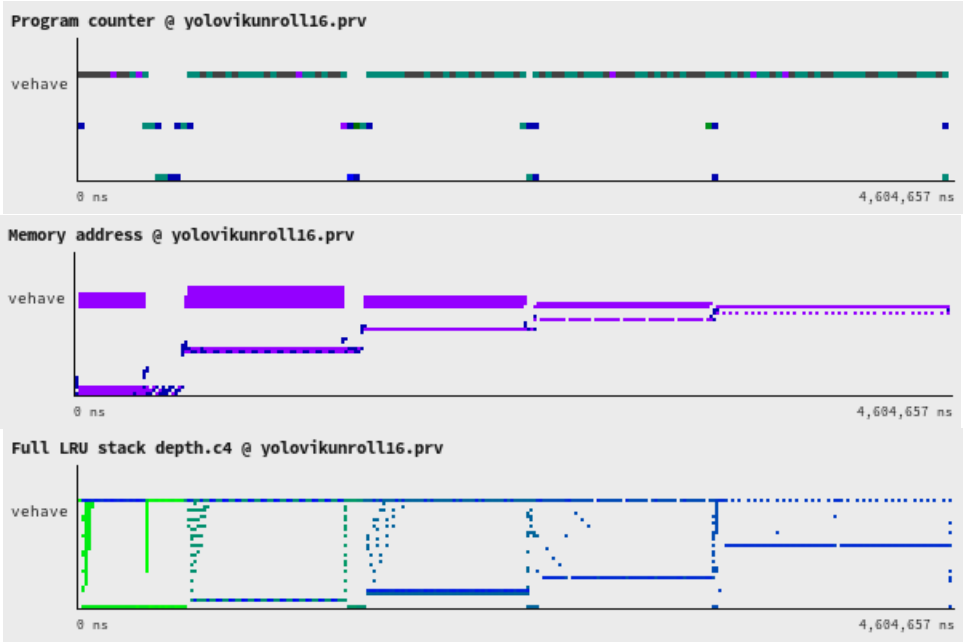
Program counter

Instruction? Order? Footprint?
Dependences? ROB size?

Memory address

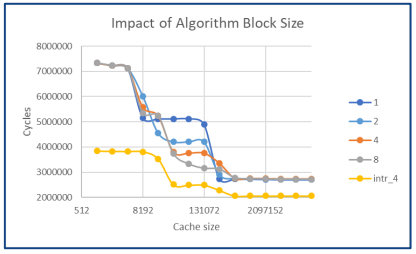
Cache management hints

LRU stack depth



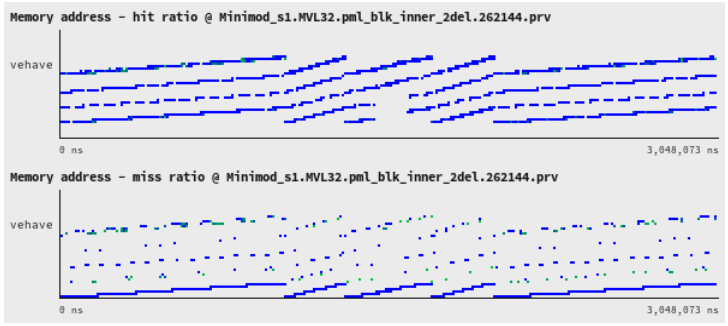
# RVV Software emulation

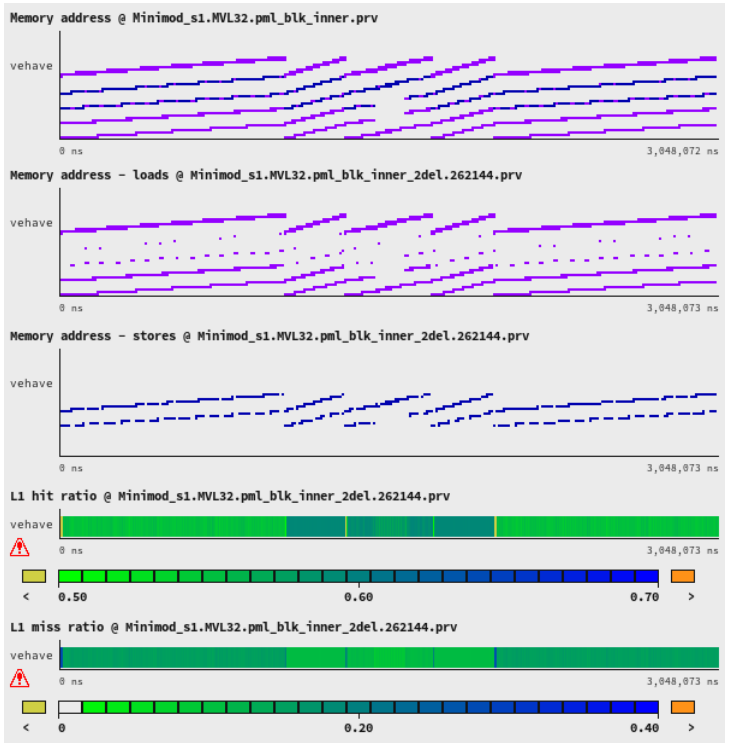
- Geophysics





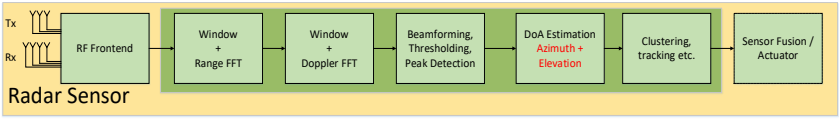
256K cache



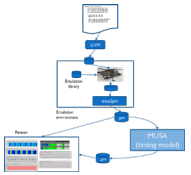



# RVV Software emulation

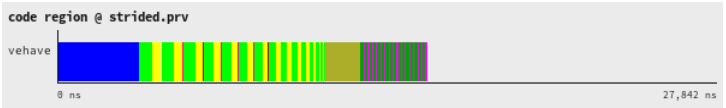
- Radar miniapp

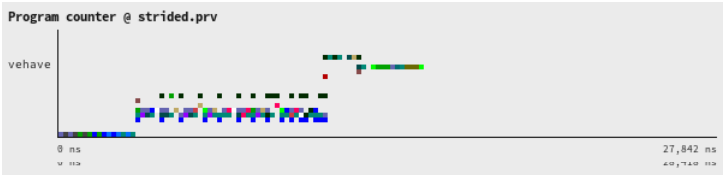


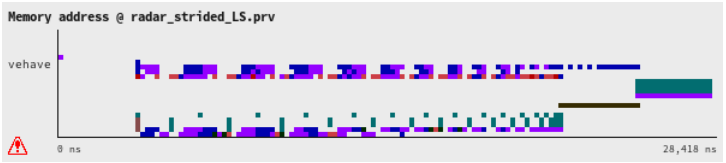

- Directives + automatic vectorization
  - Incremental way



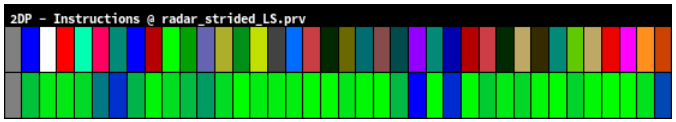
- Steering compiler optimizations
  - Complex data types
  - Indexed → strided
  - Reuse through registers
  - Avoid optimizations generating extremely short vector lengths



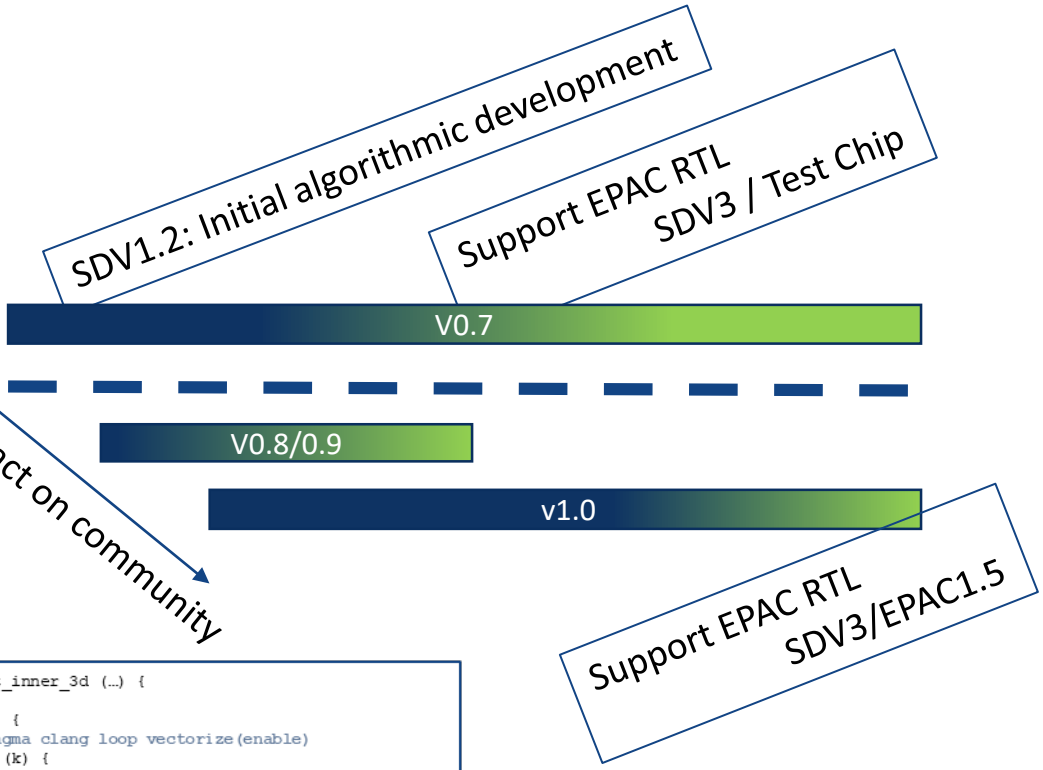




- Steering code refactoring
  - With productivity in mind !
    - Interchange, collapse, …



# RISC-V Vector extension (RVV) Compiler

- LLVM support for the evolution of the RISC-V Vector (RVV) Extension
- Intrinsics
- Autovectorization




```
void SpMV_vec(double *a, long *ia, long *ja, double *x, double *y, int nrows) {
    for (int row = 0; row < nrows; row++) {
        int nnz_row = ia[row + 1] - ia[row];
        unsigned long gvl;        // granted vector lengths
        int idx = ia[row];
        __epi_1xf64 v_a, v_x, v_prod, v_partial_res;
        __epi_1xi64 v_idx_row, v_three;
        y[row]=0.0;
        v_partial_res = __builtin_epi_vbroadcast_1xf64(0.0, gvl);
        for(int colid=0; colid<nnz_row; ) {    //blocking on MAXVL
            gvl       = __builtin_epi_vsetvl(nnz_row - colid, __epi_e64,
            v_a       = __builtin_epi_vload_1xf64(&a[idx+colid], gvl);
            v_idx_row = __builtin_epi_vload_1xi64(&ja[idx+colid], gvl);
            v_three   = __builtin_epi_vbroadcast_1xi64(3, gvl);
            v_idx_row = __builtin_epi_vsll_1xi64(v_idx_row, v_three, gvl)
            v_x       = __builtin_epi_vload_indexed_1xf64(x, v_idx_row, g
            v_prod    = __bu
            v_partial_res =
gvl);
            colid  += gvl;
        }
        y[row] += __builtin_
    }
}
```

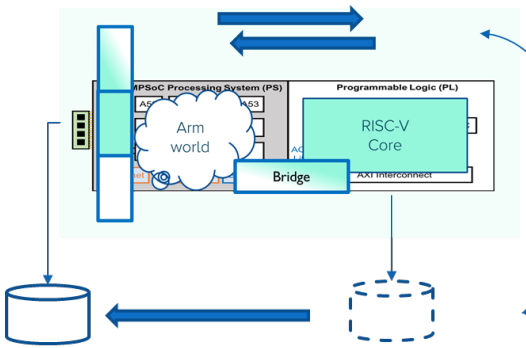
```
void target_inner_3d (…) {
  for (i) {
    for (j) {
      #pragma clang loop vectorize(enable)
      for (k) {
        lap = LAP;
        v[IDX3_1(i,j,k)] = 2.f*u[IDX3_1(i,j,k)]
                -v[IDX3_1(i,j,k)]+vp[IDX3(i,j,k)]*lap;
      }}}}
```

```
…
float complex A[n][n];
float complex temp1, temp2;
…
for (int j = 0; j < n; j++)    {
   if (x[j] != ZERO || y[j] != ZERO)  {
      temp1 = alpha * conjunction(creal(y[j]), cimag(y[j]));
      temp2 = conjunction(creal(alpha * x[j]), cimag(alpha * x[j]));
      #pragma clang loop vectorize(assume_safety)
                  for (int i = 0; i <= j - 1; i++) A[i][j] = A[i][j] + x[i] * temp1 + y[i] * temp2;
                  A[j][j] = creal(A[j][j]) + creal(x[j] * temp1 + y[j] * temp2);
   } else  A[j][j] = creal(A[j][j]);
}
```

# Heterogeneous ARM + RISC-V

- Linux Boot on  both Arm and RISC-V
  - kernel version updates
  - Tracking mainline, and contributing to ongoing patch testing and review (eg. SV48, huge pages)
- OpenMP offloading
  - Asynchronous calls (via service thread pool)
  - Thread teams
- Reverse offloading
  - access to host-side I/O devices
  - Works for Linux process on RISC-V side
  - WIP for offloaded tasks

FORTH

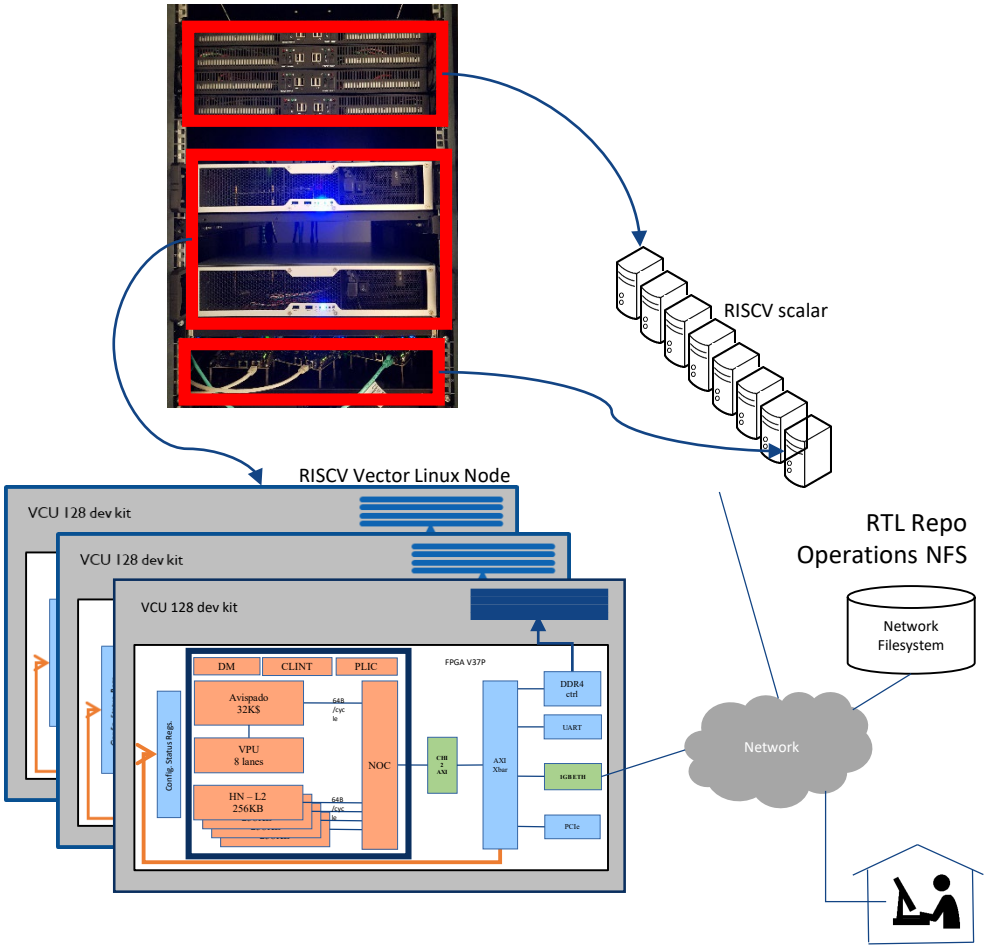



```
#on RISC-V side
$mkfs.ext4 -b 4096 /dev/vda
$mount -t ext4 /dev/vda /mnt/scratchfs
```

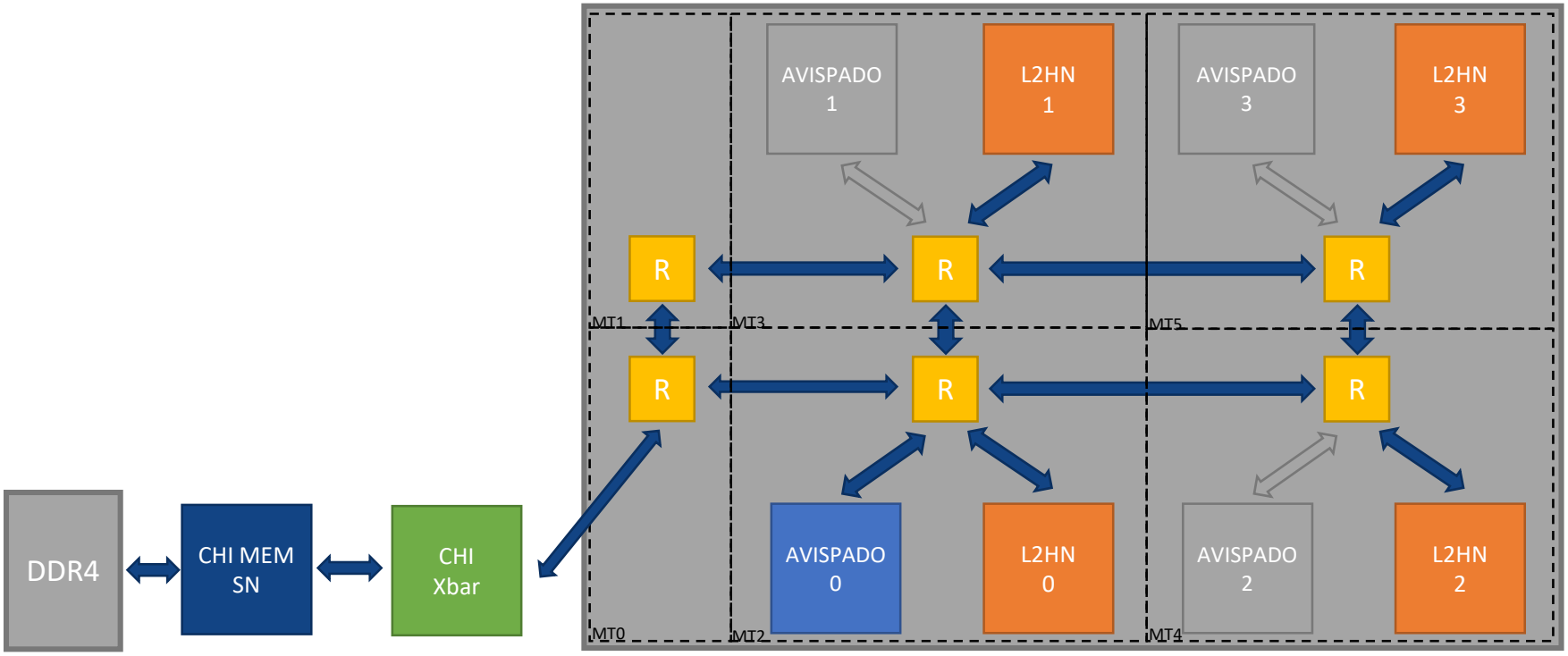
```
void main (…)
{ …
   gemTarget(A, B, C, size);
… }

void gemTarget(double *A, double *B, double *C, long S) {
  #pragma omp target map(to:A[0:S*S], B[0:S*S], S) \
                     map(from: C[0:S*S])
  {    for(int i = 0; i < size; i++)
         for(int j = 0; j < size; j++)
           for(int k = 0; k < size; k++)
             C[i*size + j] += A[i*size + k]*B[k*size + j];
       fp = fopen ("/mnt/scratchfs/output_matrix.txt", "w+");
       for (int i=0; i<size*size; i++) fprintf(fp, "%lf ", C[i]);
  }
}
```

# RVV @ FPGA & ecosystem

- HPC software stack @ Commercially available RISC-V platforms
  - SLURM, MPI, OpenMP, BSC tools, RVV software emulator
- EPI SDV platforms
  - Booting Linux
  - Test user codes @ real RTL
  - Give to EPI partners and external users early access to EPI technology
    - Two step procedure
- Holistic CI/CD framework
  - HW & SW
  - Functionality & performance

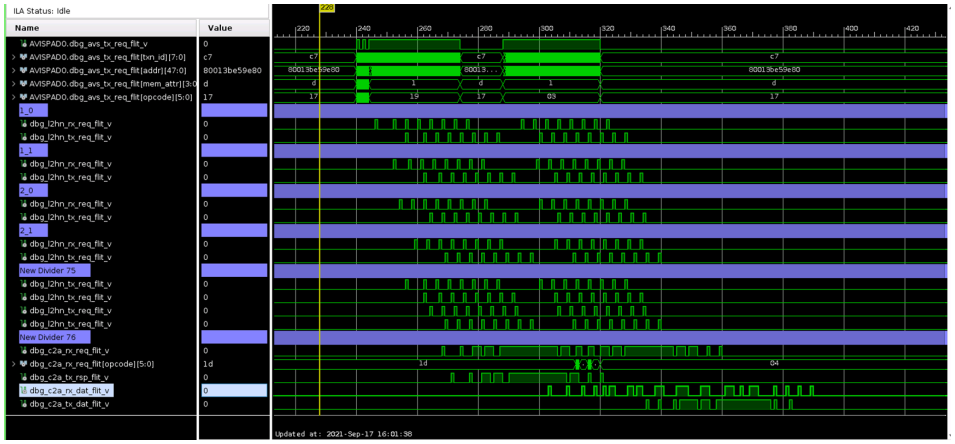


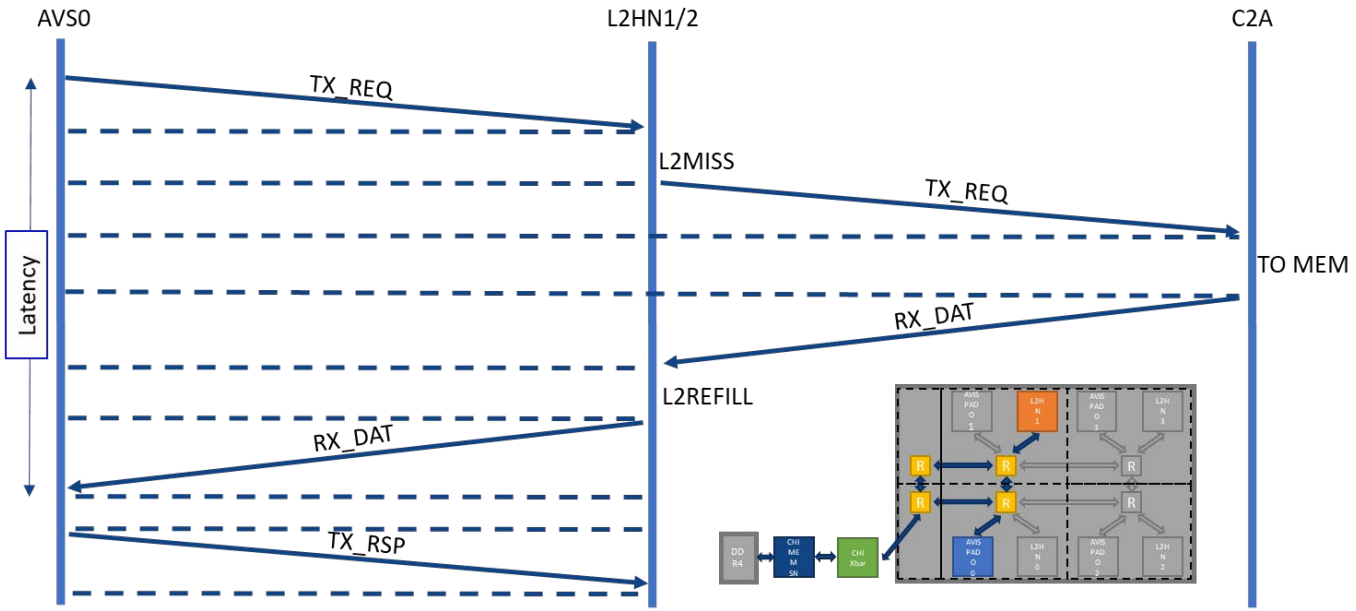
# Single Core SDV



# Memory subsystem

- Mapping
- Protocol
  - DMT or not
- Memory dilation IP
- NoC
  - Injection, credits
  - hops/contention
- Pipelining
  - Bubbles?



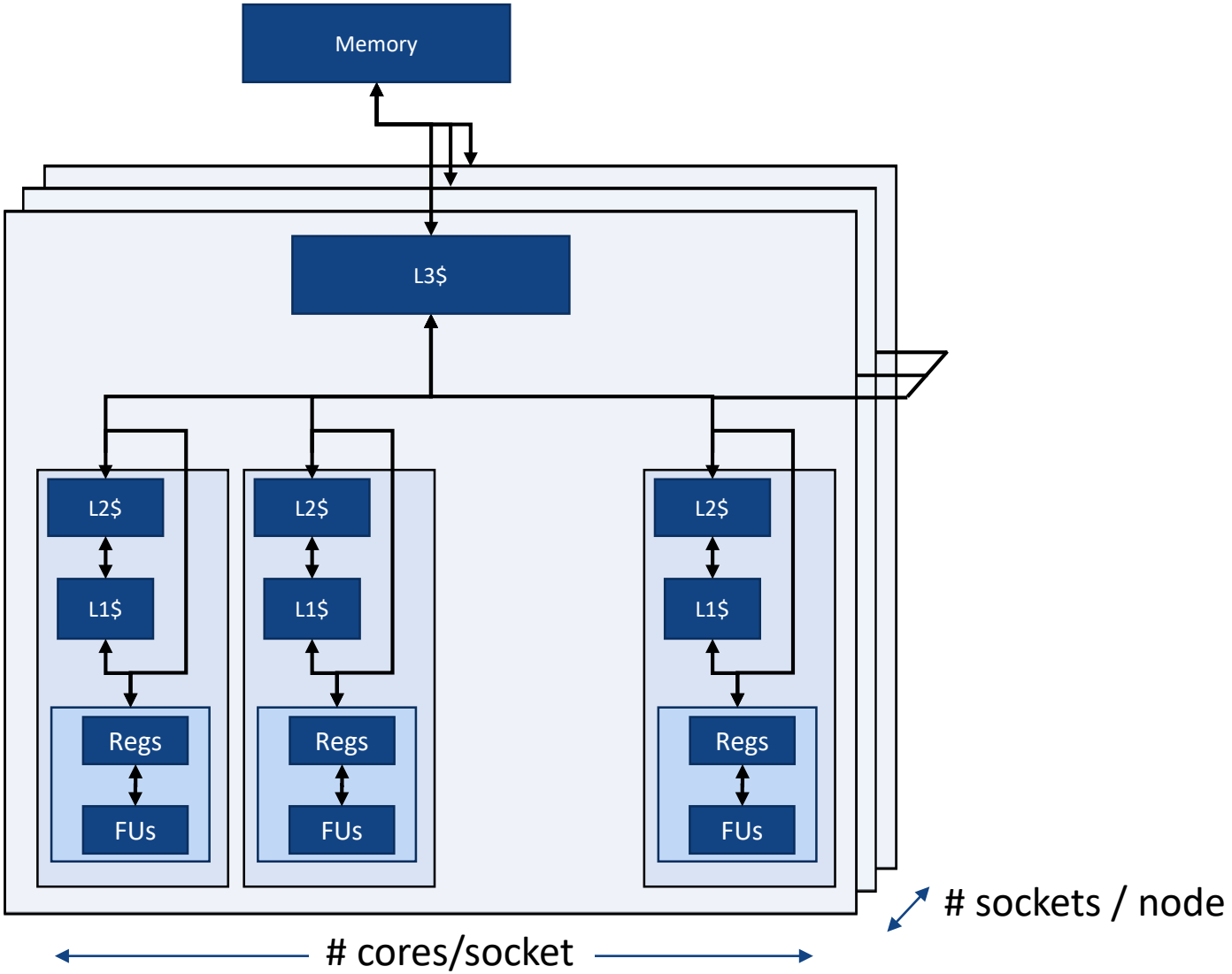


# Continuous Co-design

- Comparing → Insight
  - Architectures
    - Core – Core / Socket – socket / Node – Node
    - Capacities
    - Bandwidths
    - ISO frequency, (power, area, cost, …)
  - Applications
    - Algorithmic refactoring
    - Programmability

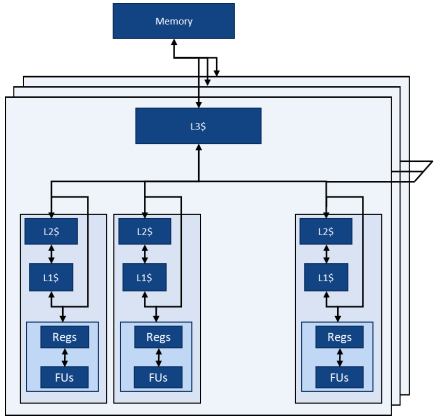
- Insight → “Design” Impact
  - Application
  - Compiler
  - Architecture

# “General Architecture”



# Comparisons …

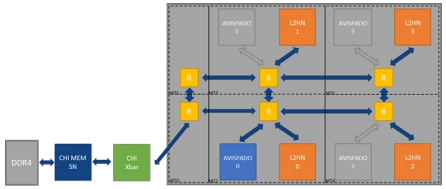
- Architecture
  - Core – Core / Socket – socket / Node – Node
  - Capacities
  - Bandwidths

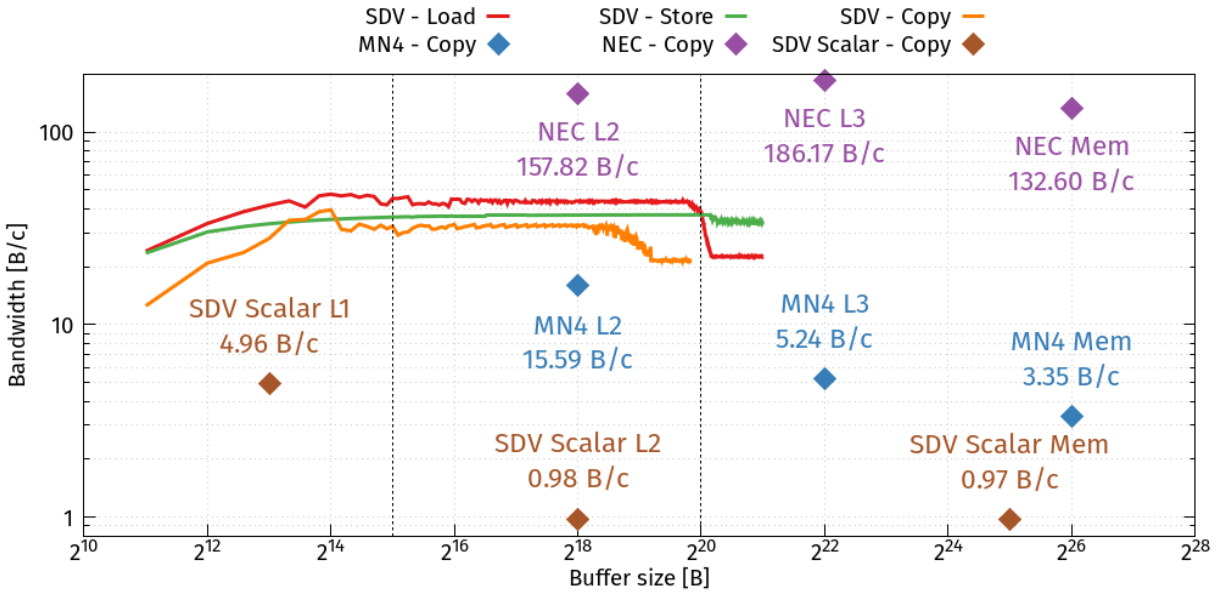



| | | SDV | SX-Aurora | CTE-Arm | MareNostrum4 |
|---|---|---|---|---|---|
| | System integrator | EPI | NEC | Fujitsu | Lenovo |
| | Core architecture | RISC-V | SX VE | Armv8 | Intel x86 |
| | IO / OoO | IO | OoO | OoO | OoO |
| | SIMD/vector extension | V | SX Vector | SVE | AVX512 |
| | MAXVL | 256 | 256 | 8 | 8 |
| | CPU name / model | EPAC | VE10B | A64FX | Xeon Platinum 8160 |
| | Frequency [GHz] | 0,05 | 1,4 | 2,2 | 2,1 |
| | Turbo Boost | - | Disabled | Disabled | Disabled |
| | Simultaneous Multi-Threadin | - | Disabled | Disabled | Disabled |
| | Sockets / node | 1 | 1 | 1 | 2 |
| | Cores/socket | 1 | 8 | 48 | 24 |
| | Core / node | 1 | 8 | 48 | 48 |
| | #Vector functional units | 8 | 96 | 16 | 16 |
| | DP Max / core [GFlop/s] | 0,8 | 268,8 | 70,4 | 67,2 |
| | DP Max / node [GFlop/s] | 0,8 | 2150,4 | 3379,2 | 3225,6 |
| Capacity | Memory / node [GB] | 4 | 48 | 32 | 96 |
| Capacity | Shared L3[MiB] | 1 | 16 | 32 | 33 |
| Capacity | Private L2 [KiB] | 0 | 256 | 0 | 1024 |
| Capacity | Private L1[KiB] | 32 | 32 | 64 | 32 |
| Capacity | Physical Registers [KiB] | 80,25 | 512,88 | 9,13 | 11,9 |
| Capacity | Total Node [KiB] | 1136,25 | 22791,00 | 36278,00 | 118843,20 |
| Capacity | Total per core [KiB] | 1136,25 | 2848,88 | 755,79 | 2475,90 |
| Capacity | Total per Flop [B/F] | 71,02 | 14,84 | 23,62 | 77,37 |

# Observed Memory Bandwidth

- Single core Load, Store, Copy
- Comparison to scalar and other architectures
- Comparison to their architectural peak

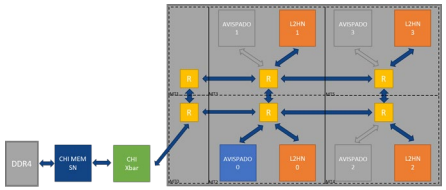


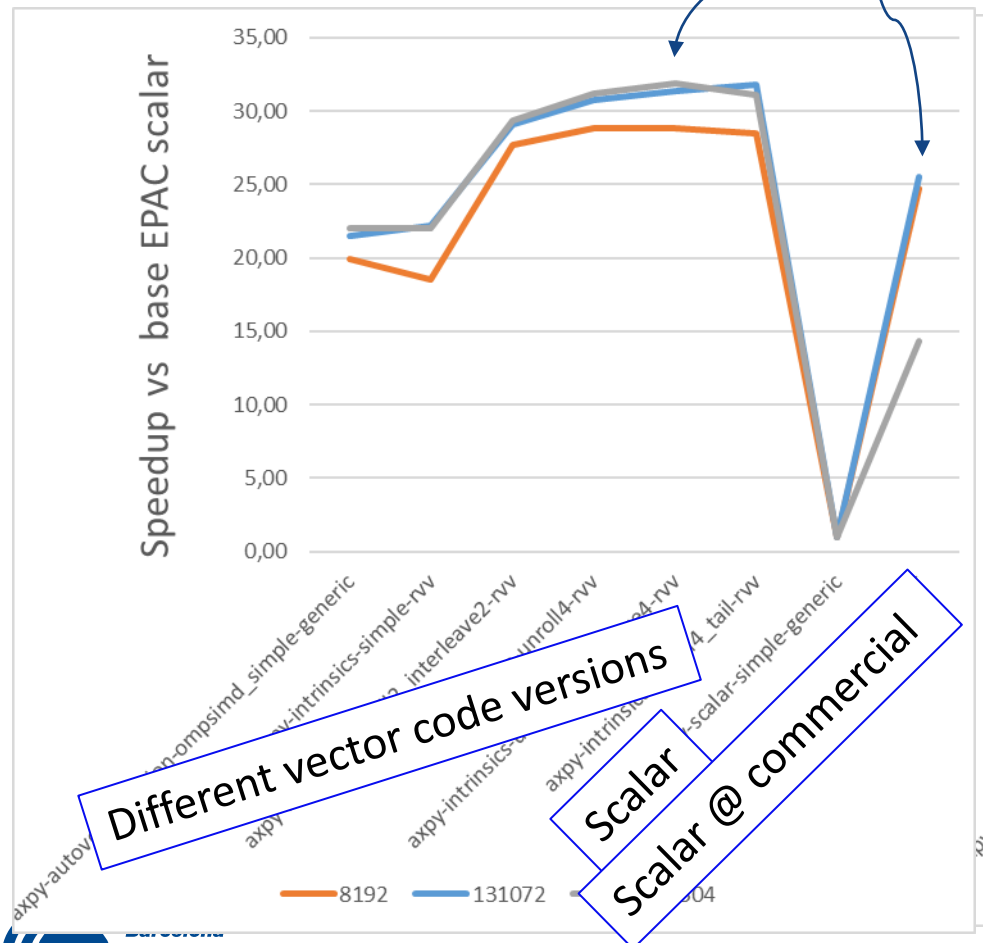


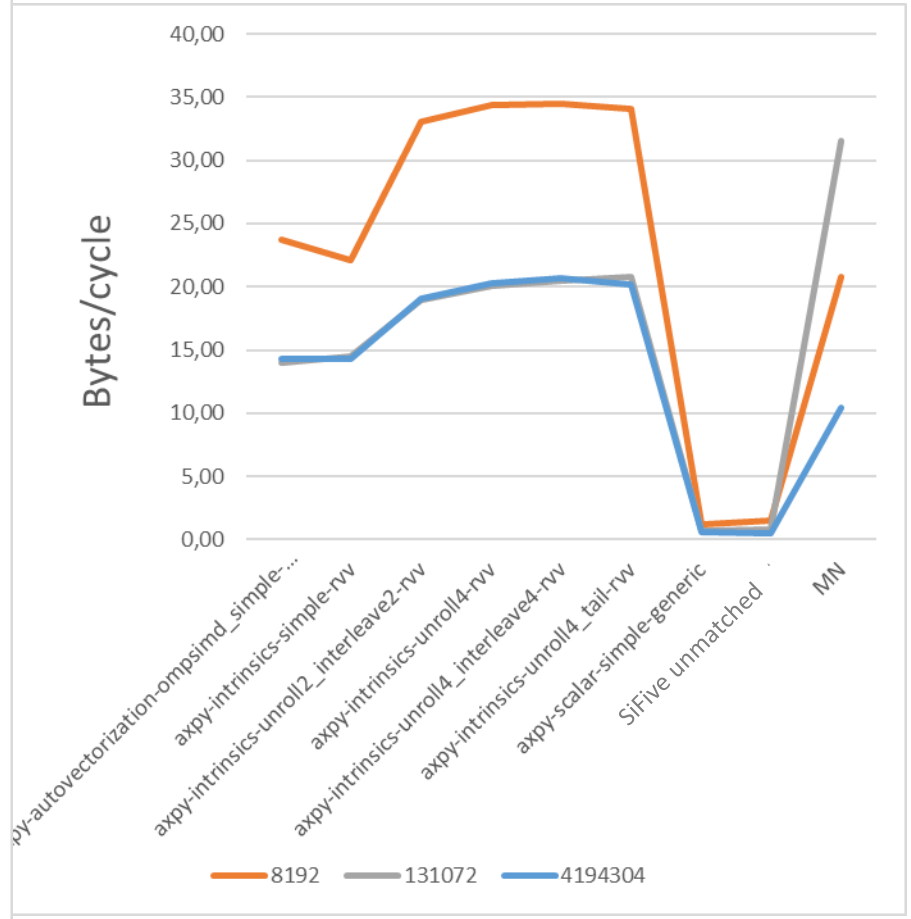
# Continuous Co-design

Axpy

FPGA 50 MHz vs ASIC 1 GHz

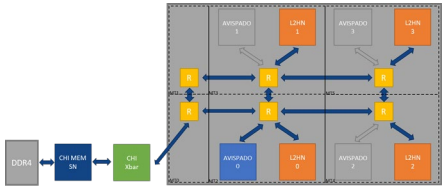


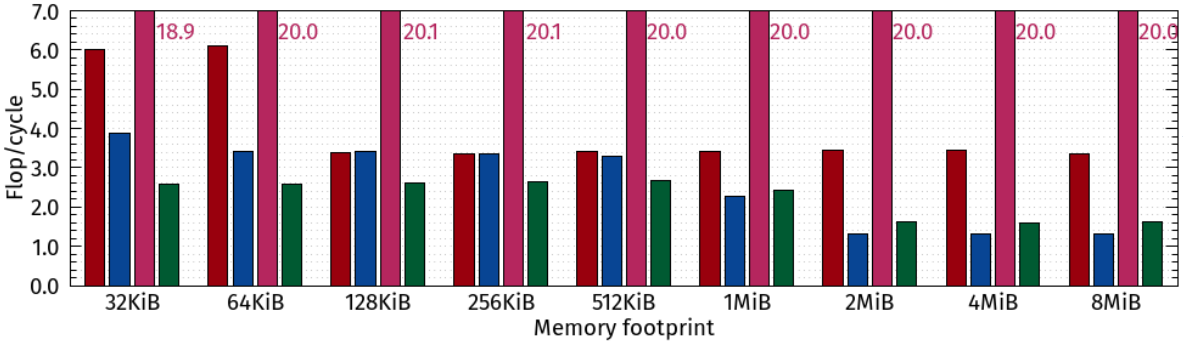


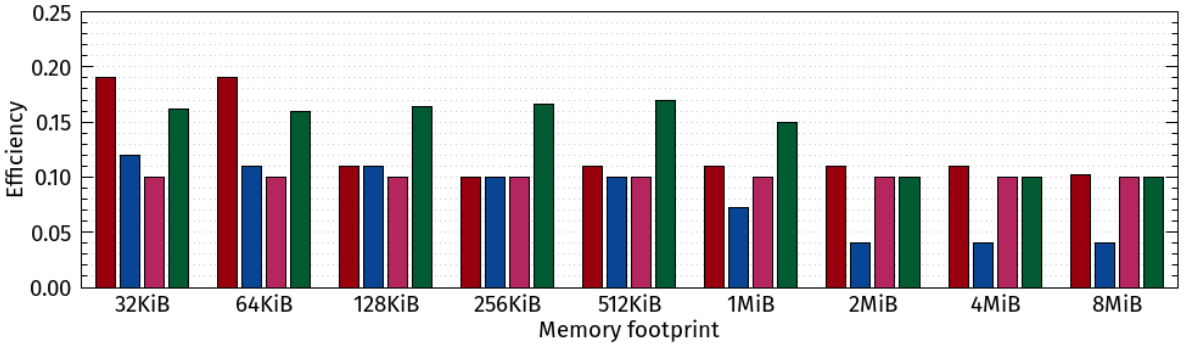
# Continuous Co-design

- Daxpy



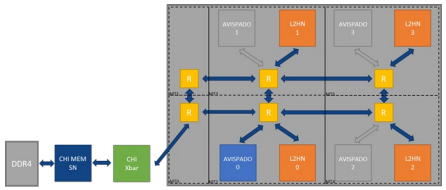


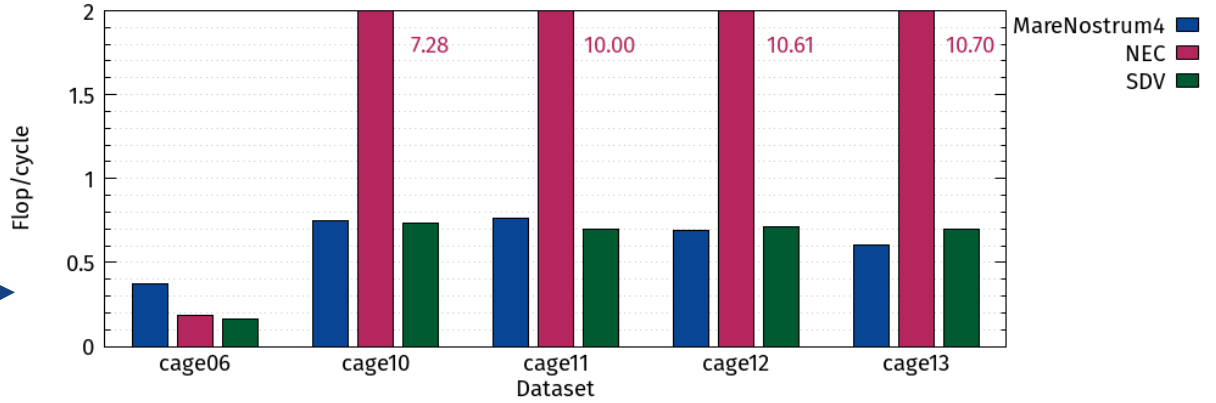

Ratio ?



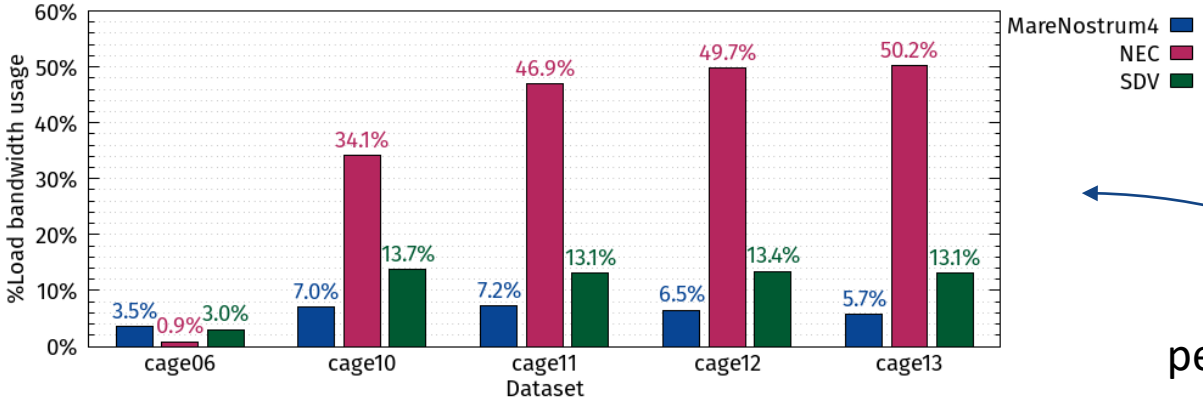
# Continuous Co-design

- SpMV
  - MKL / NLC / Ellpack based implementation





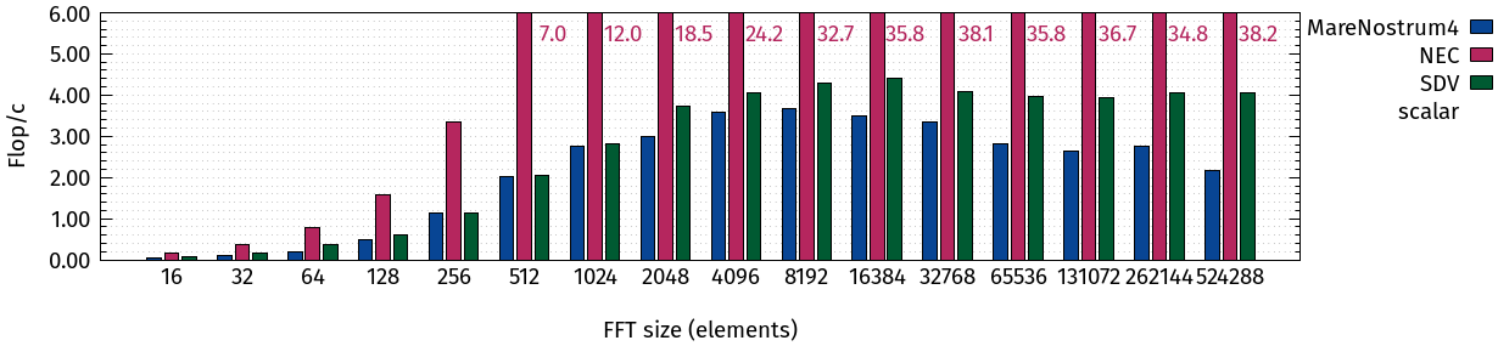

Scalar performance ?

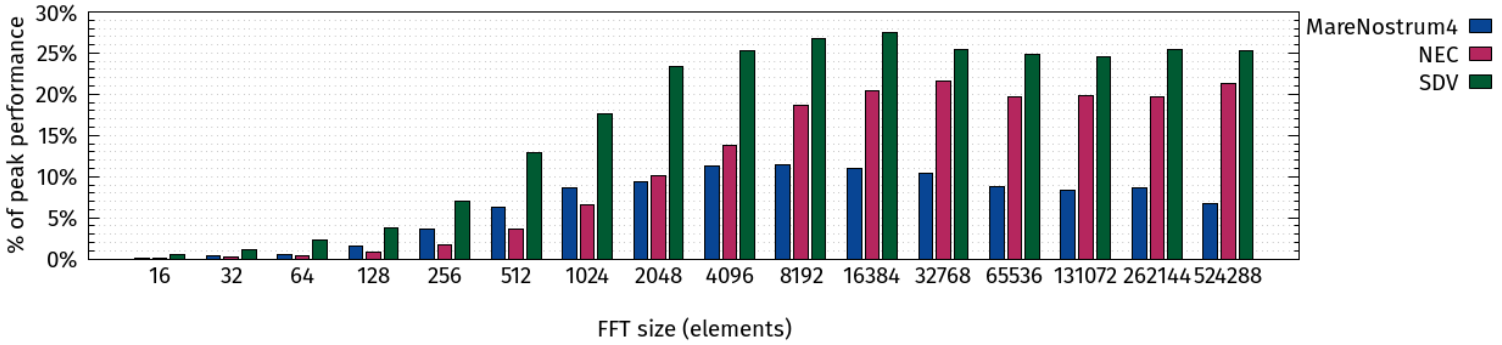



Gather performance ?

# Continuous Co-design

- FFT

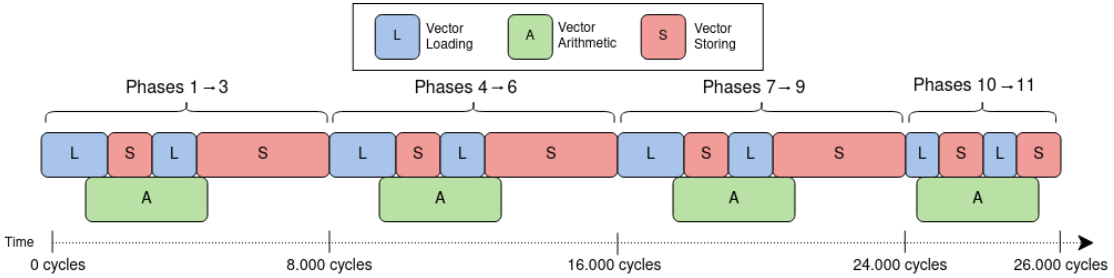


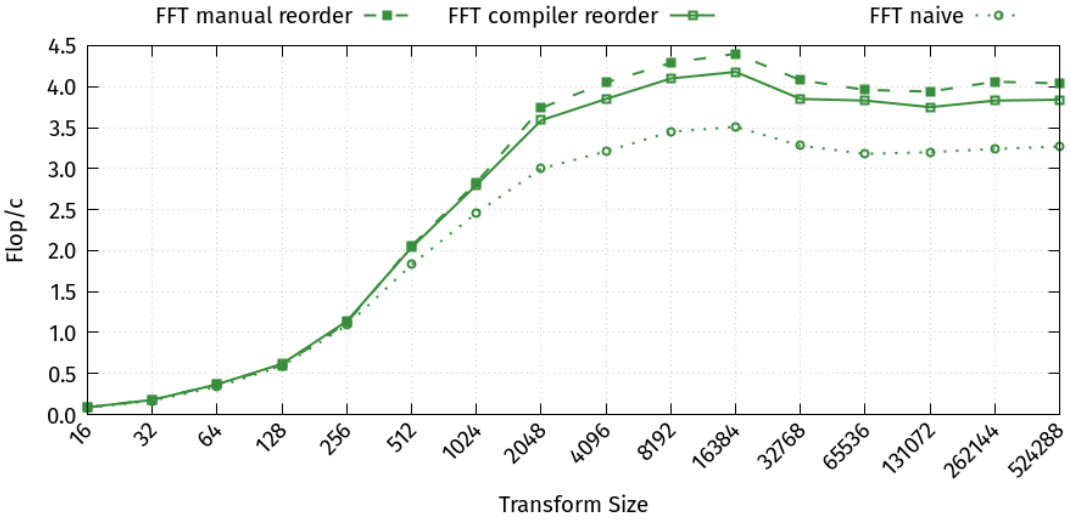



# Continuous Co-design

- FFT

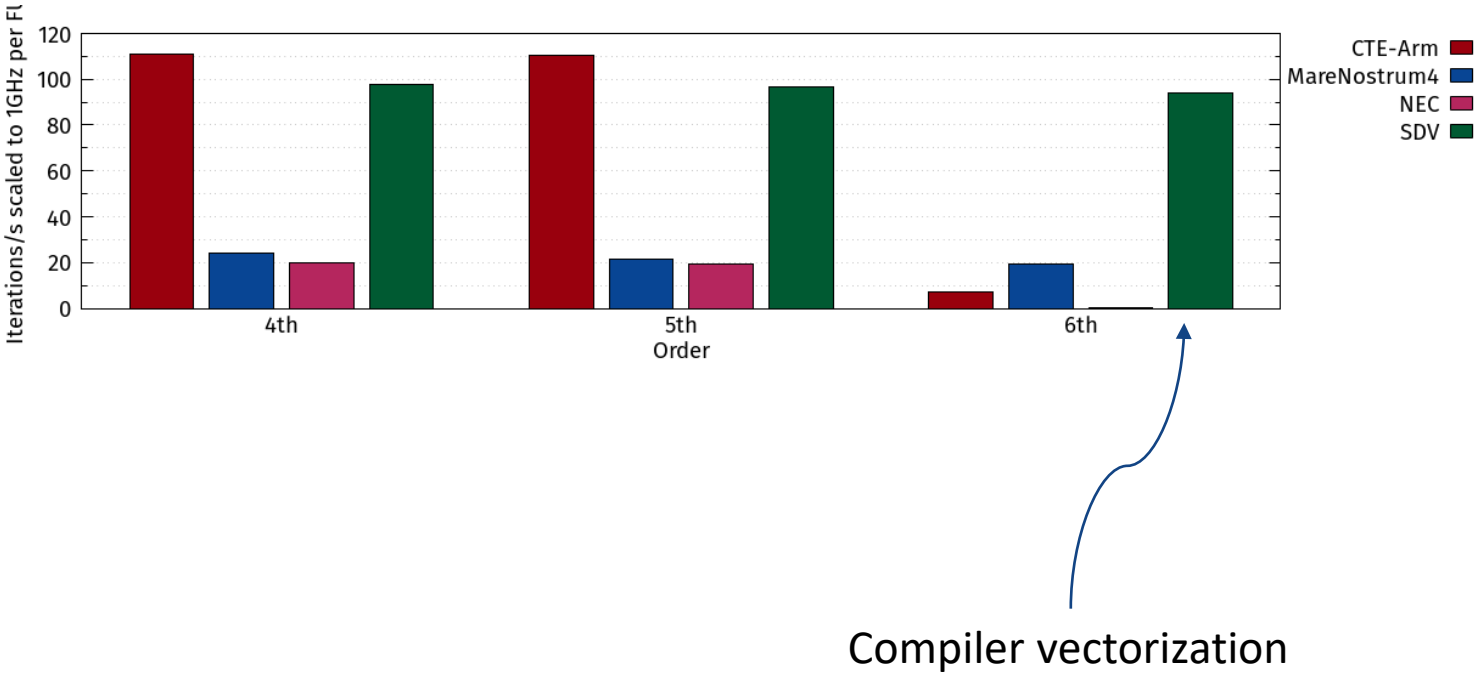
Instruction scheduling



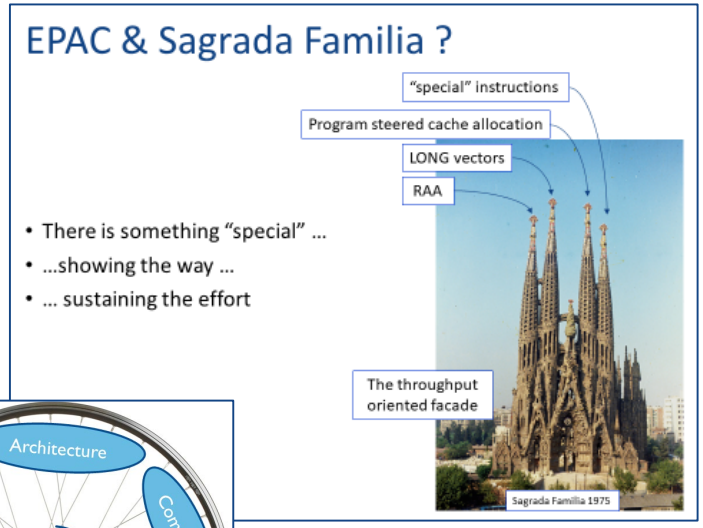
# Continuous Co-design

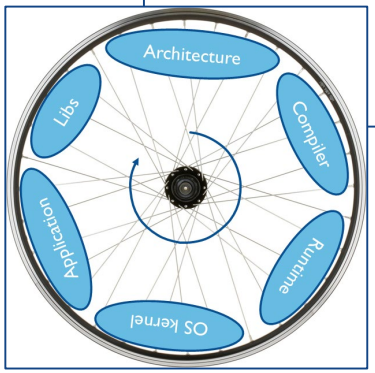
- HACC



# The importance of a vision


- Holistic throughput oriented vision based on long vectors and task based models
- Hierarchical concurrency and locality exploitation
  - Not massive concurrency at a given level
  - Push behaviour exploitation to low levels
- Co-ordination between levels
- Make it all look very close to classical sequential programming to ensure productivity




RISC-V

European Processor Initiative

https://repo.hca.bsc.es/gitlab/epi-public/

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

# Thanks