Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

The RISC-V vector processor in EPI

Prof. Jesús Labarta

BSC & UPC

Barcelona, September 27rd 2021

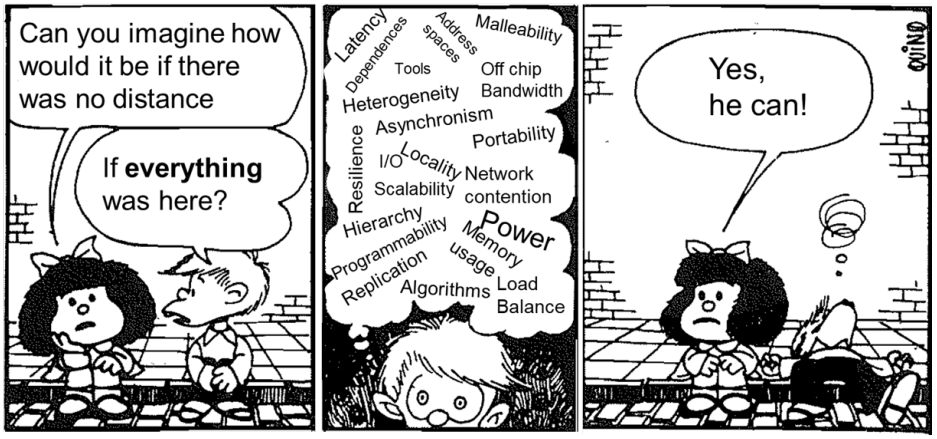
Age before beauty (disclaimer)

- 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

El abuelo cebolleta ataca de nuevo

EPI Objectives

- Components (low power microprocessor technologies) **…**
  - ARM based SoC
  - RISCV based accelerator
- … to be combined to target
  - HPC
  - HPDA
  - Emerging
    - Automotive
    - …

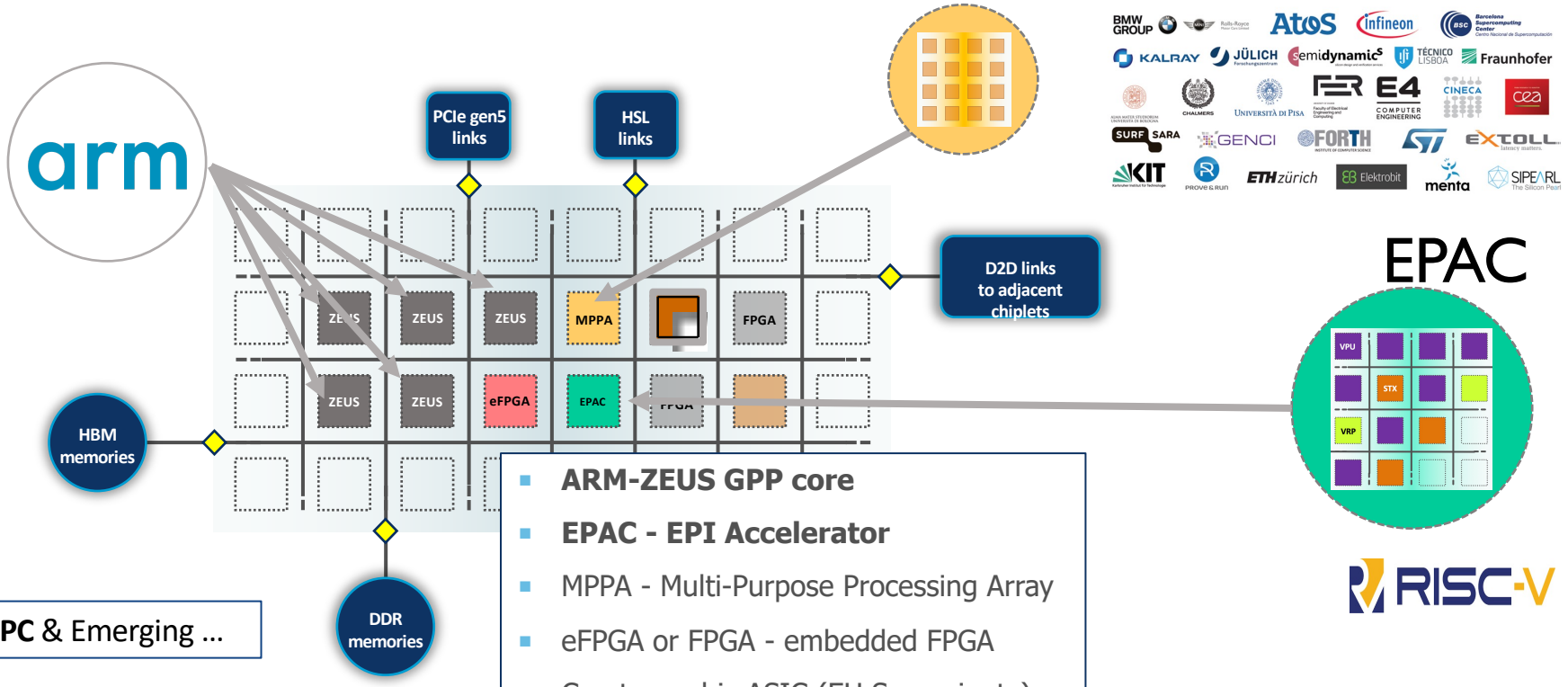


- Co-design role:
  - Application characterization. Identification of fundamentals !!!

Three streams

- General purpose
  - ARM SVE
  - BULL: System integrator → chip integrator
    - Spinoff → SiPearl
- Accelerator
  - RISC-V
  - EU design: BSC, CEA, Chalmers, ETHZ, EXTOLL, E4, IST, FORTH, FhG, Semidynamics, UNIBO, UNIZG
- Automotive
  - Infineon, …

Overall architecture



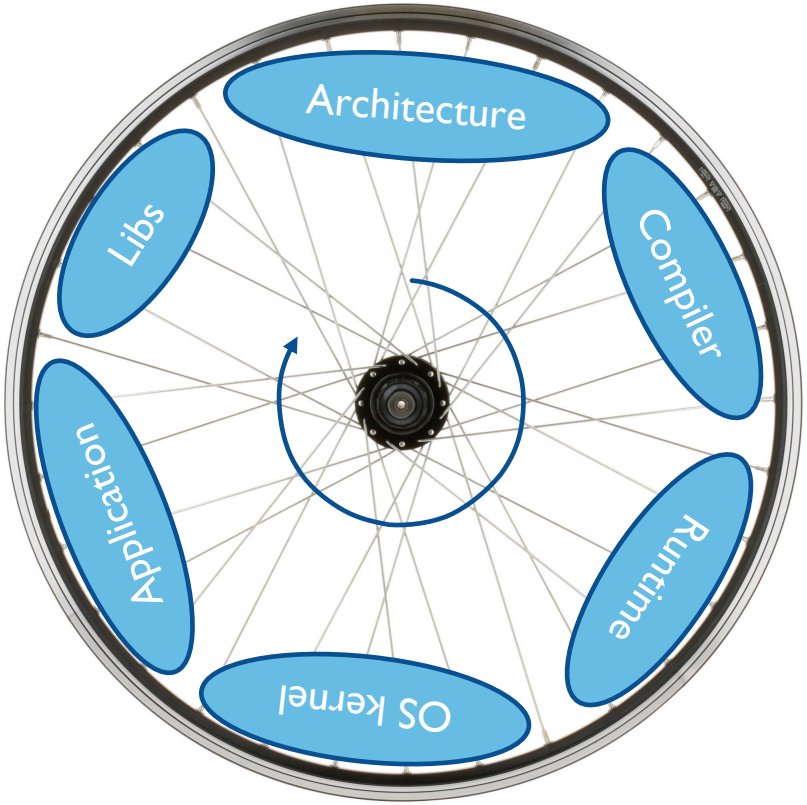
- **ARM-ZEUS GPP core**
- **EPAC - EPI Accelerator**
- MPPA - Multi-Purpose Processing Array
- eFPGA or FPGA - embedded FPGA
- Cryptographic ASIC (EU Sovereignty)

HPC & Emerging …

Towads “Exascale”

Holistic Co-design

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



Best place to address an issue

Fundamentals

Balance

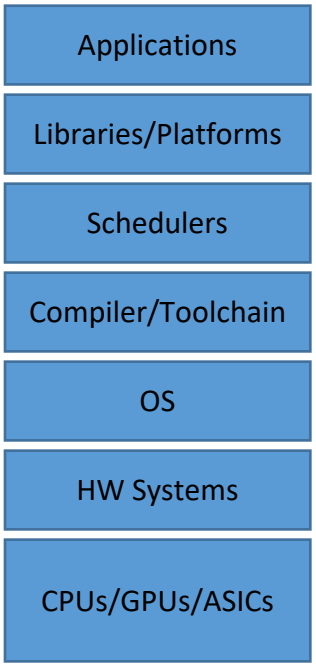
Mindset

Productivity

Efficiency

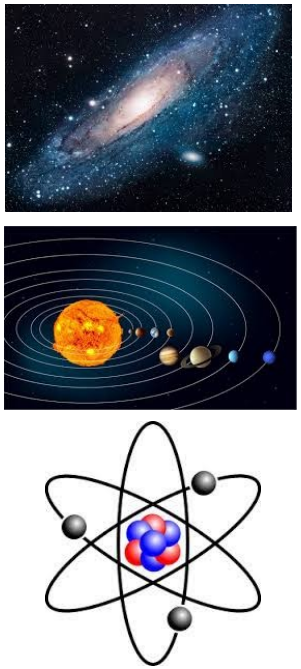
The need/ideal of a vision

- A unified theory
  - Technical
  - Economy / Politics



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

"Of fundamental principles"
"Steered by detailed insight"



Still not there …
… still not able to communicate ☹

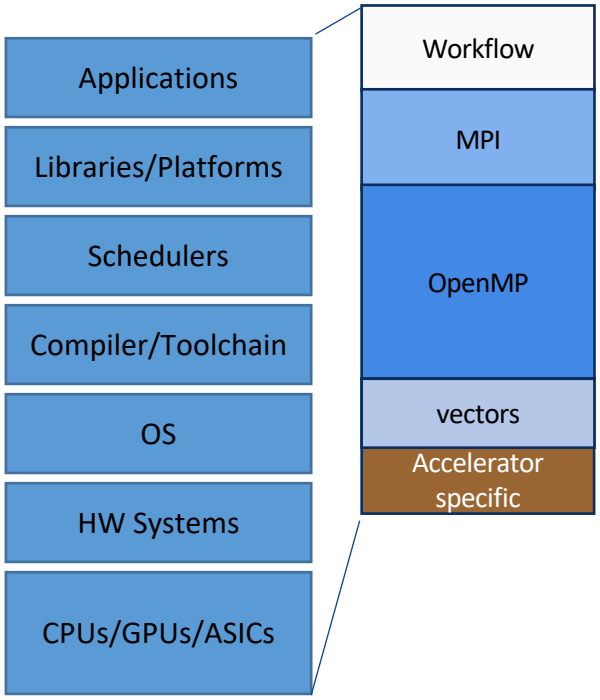
principles 

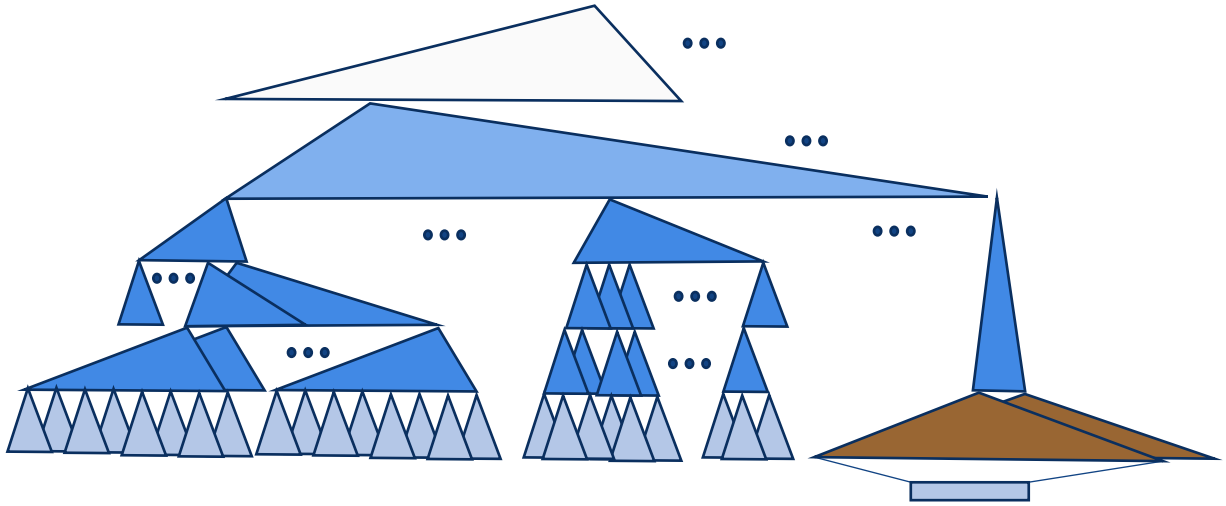
- Balanced hierarchy
- Throughput oriented: asynchrony and overlap
- Malleability and coordinated scheduling
- Homogenizing heterogeneity
- Detailed analysis and insight on behavior

Balanced hierarchy

Expression & exploitation of Parallelism

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





RISC-V

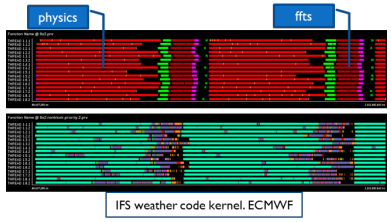
Long vectors. 8 lanes per core
"Limited" number of "general purpose" control flows within tile

Latency → Throughput: asynchrony and overlap

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

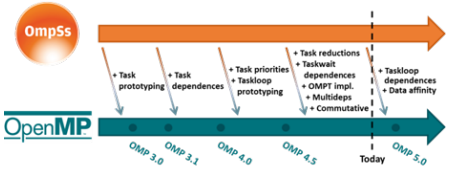
Interoperability MPI + OpenMP

- Taskify MPI calls

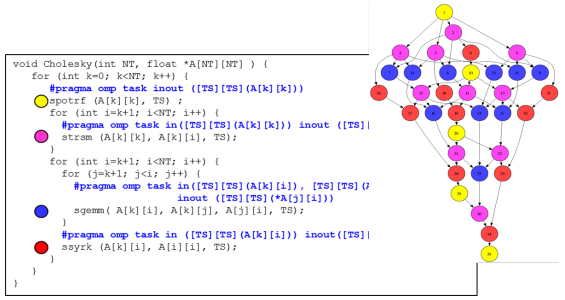


Task based models

- Single mechanism
  - Concurrency
  - Locality & data management



Task based computational workflows

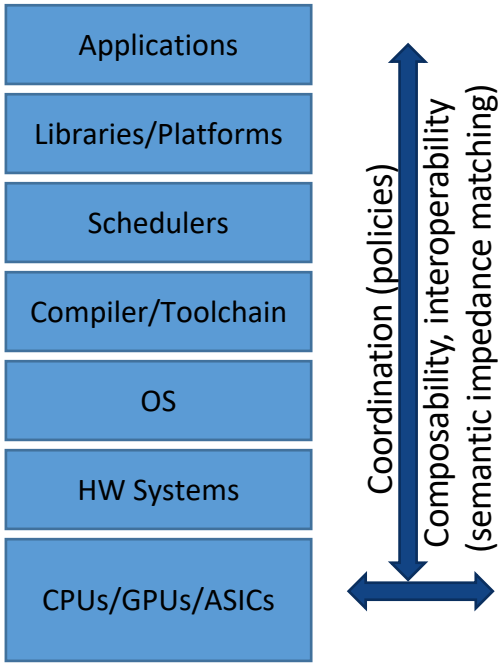


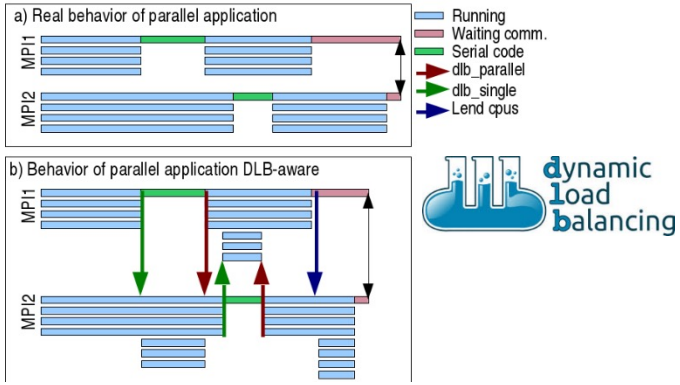
```
void Cholesky(int NT, float *A[NT][NT] ) {
  for (int k=0; k<NT; k++) {
    #pragma omp task inout ([TS][TS](A[k][k]))
    spotrf (A[k][k], TS) ;
    for (int i=k+1; i<NT; i++) {
      #pragma omp task in([TS][TS](A[k][k])) inout ([TS]
      strsm (A[k][k], A[k][i], TS);
    }
    for (int i=k+1; i<NT; i++) {
      for (j=k+1; j<i; j++) {
        #pragma omp task in([TS][TS](A[k][i]), [TS][TS](A
                inout ([TS][TS](*A[j][i]))
        sgemm( A[k][i], A[k][j], A[j][i], TS);
      }
      #pragma omp task in ([TS][TS](A[k][i])) inout([TS]
      ssyrk (A[k][i], A[i][i], TS);
    }
  }
}
```

Long vectors

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

Malleability & Coordinated scheduling





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];
}
```

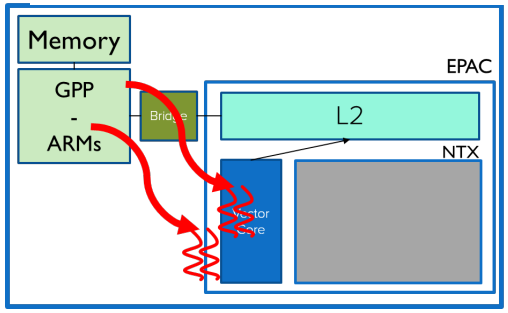
VLA helps homogenize Heterogeneous Performance

- ~ Big – Little cores, ….

Nested tasked/workshared

OpenMP

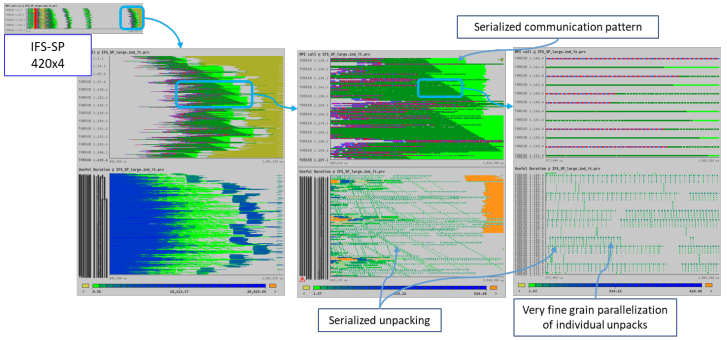
- Offload regular OpenMP



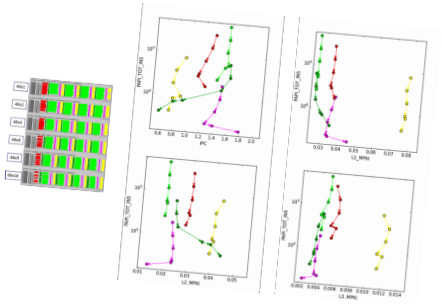
- HW support: IO coherence

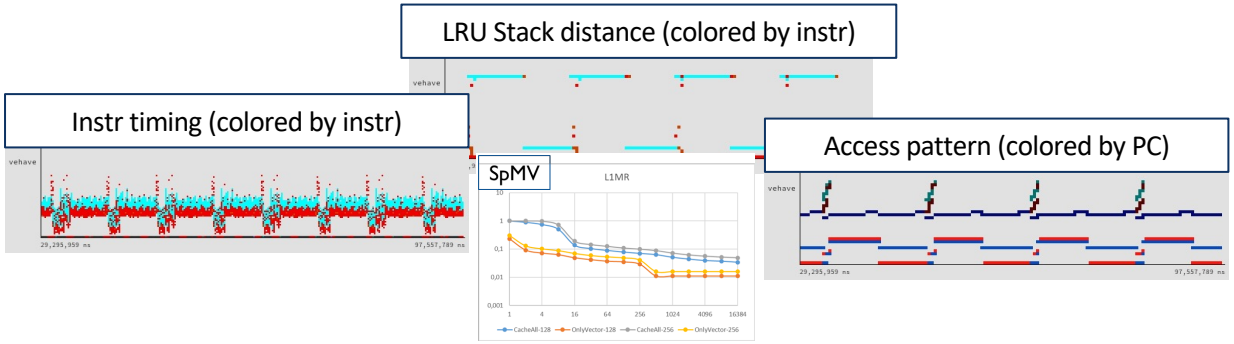
Detailed analysis and Insight on behavior

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



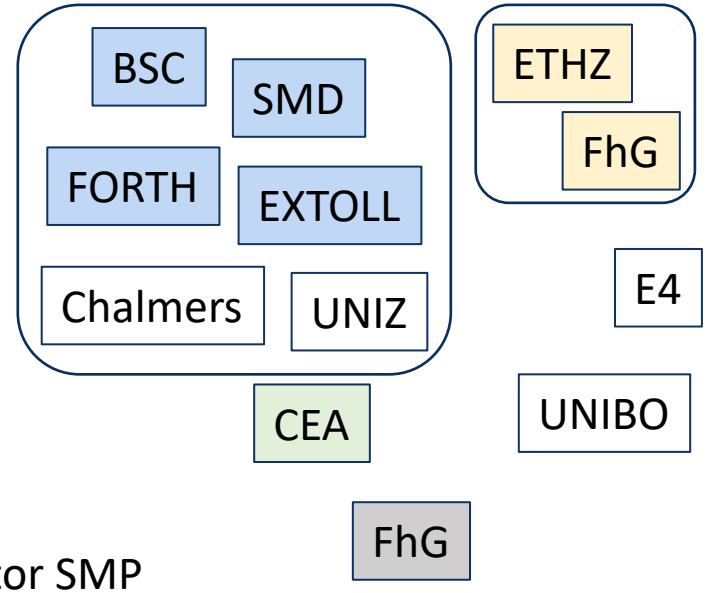
https://pop-coe.eu/





Visions and collaborations

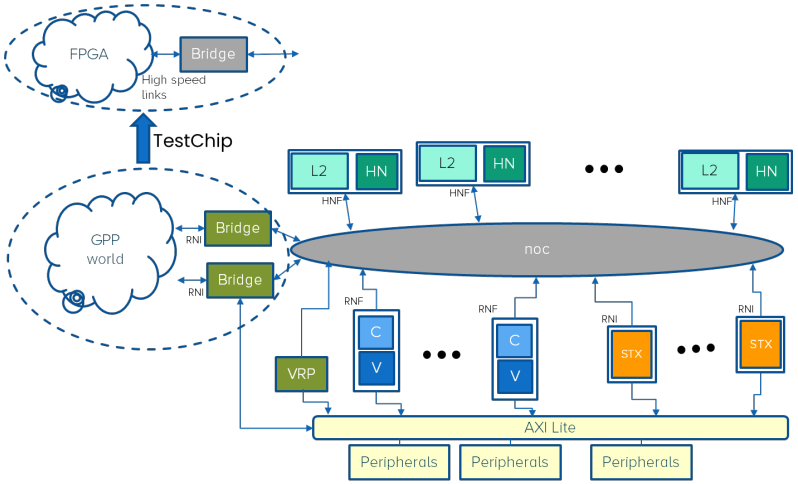
- STX:
  - Specific Accelerator devices
    - AI
    - Stencil
- RVV
  - ISA is important, RISC-V Vector
  - “Accelerator”
    - Easier entry, focus
    - → Standard self hosted, general purpose vector SMP
- VRP:
  - Extended precision arithmetic



EPAC architecture

RVV

- RV64GCV (→ 8x)
  - 2 way in order core
  - Decoupled VPU
    - 8 lanes
    - Long vectors (256 DP elements)
  - L1 - MESI coherency
- CHI interface NoC
  - 1 line / cycle
- L$2: 256KB/module
  - Allocation control mechanisms
- No in tile L$3



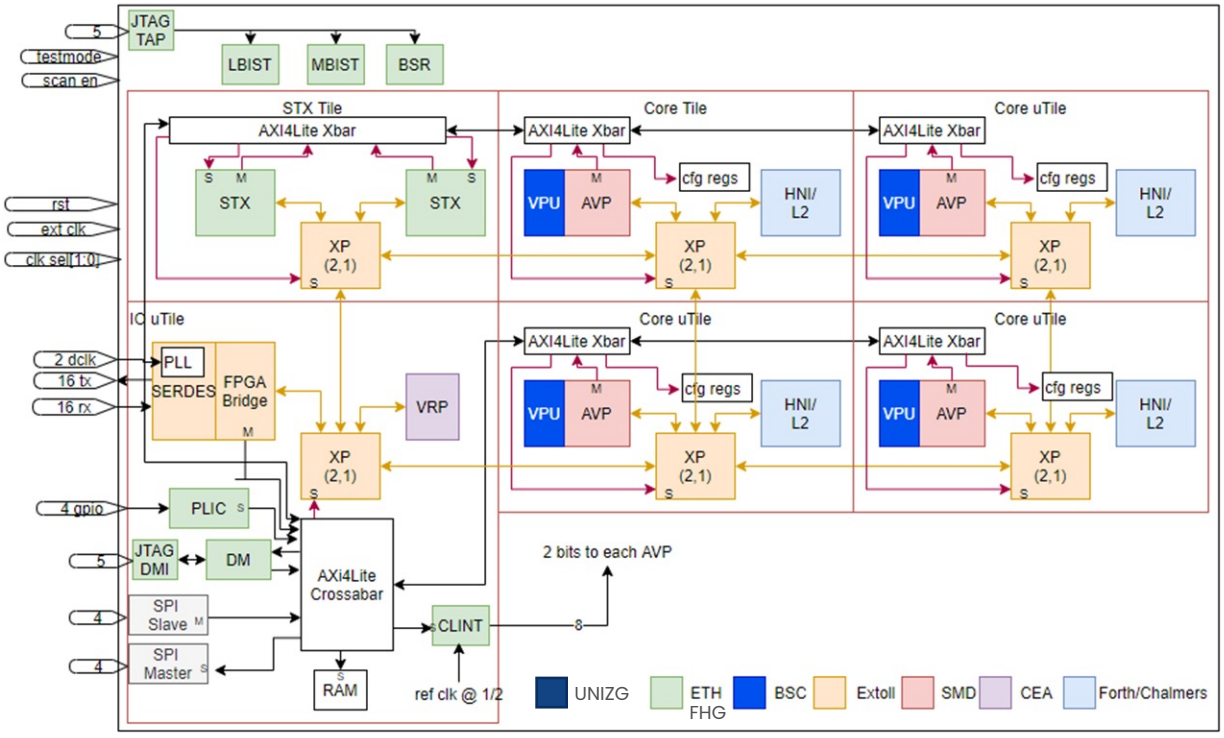
STX

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

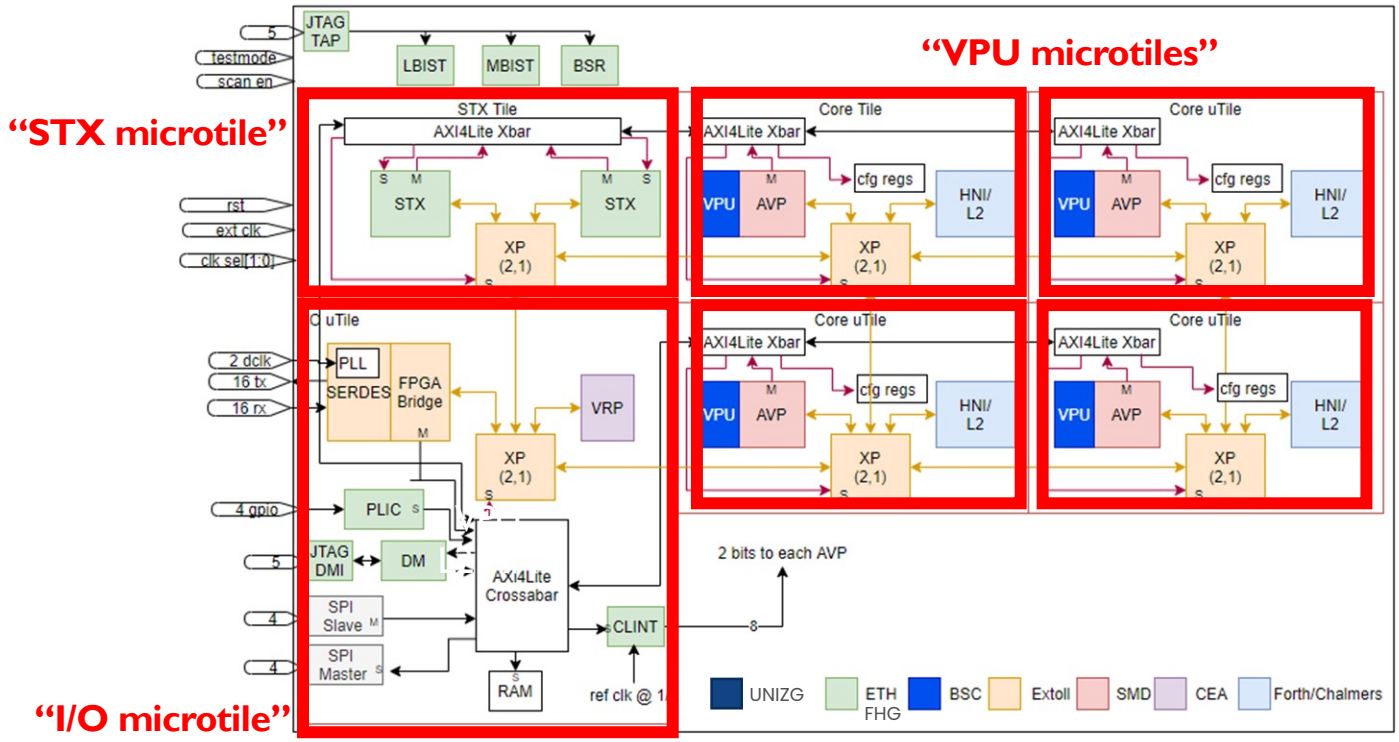
VRP

- Variable precision processors

EPAC Test Chip Block Diagram

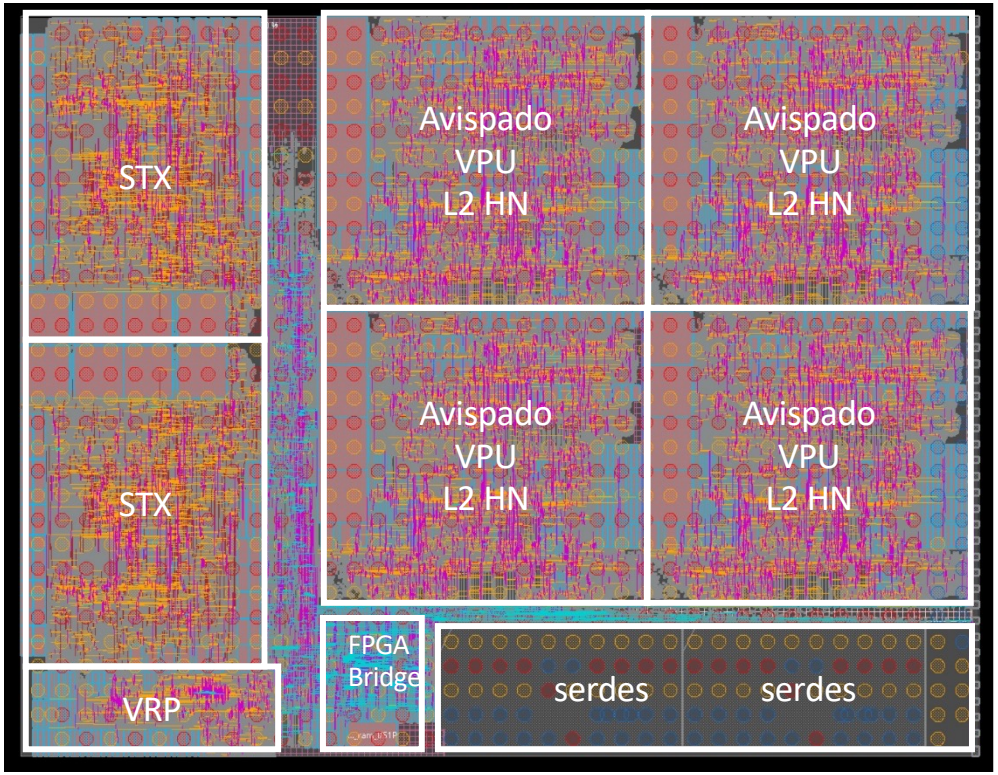


EPAC Test Chip Block Diagram



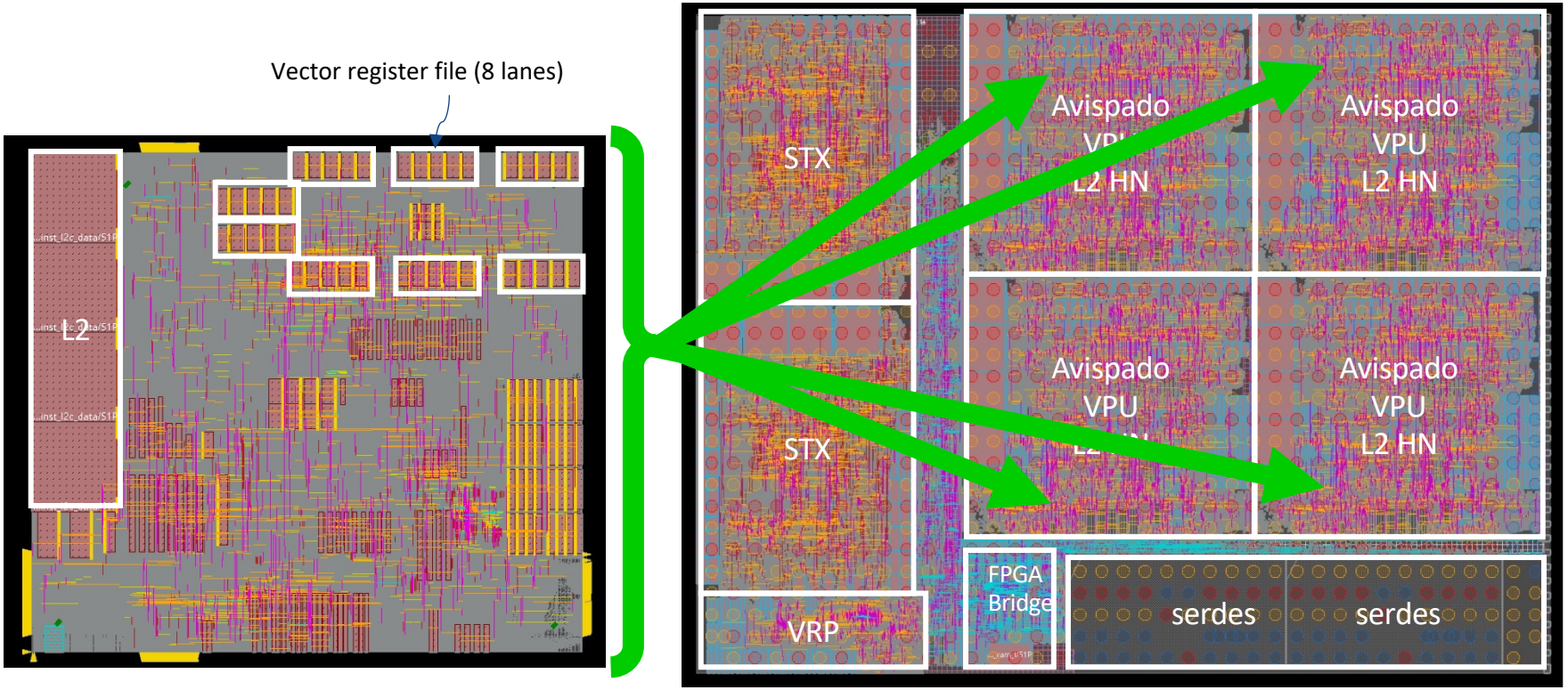
Physical design

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



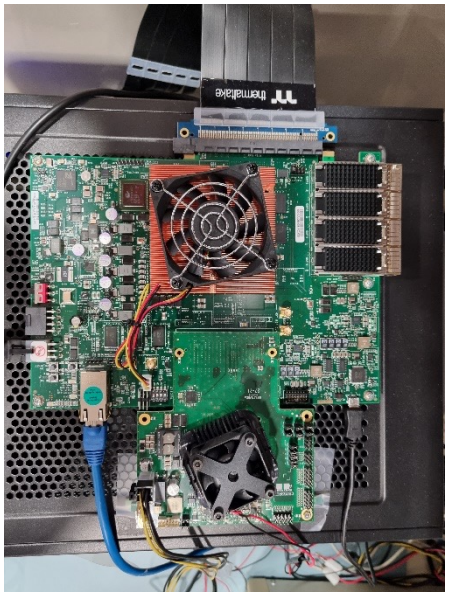
Physical design

- 4 “VPU microtiles”
  - 2.517 mm2 each



Alive

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



```
carv@bouklas: ~/Desktop/bringup_20210916/tools_new/bringup_20210917
EPAC JTAG Console Client 0.1
Connecting to JTAG Console [0] ...
Press CTRL+A for exit

-----------------------------------
| Welcome to EPAC TC Bring-Up Shell |
-----------------------------------
epac@nikitas$ help
help            Prints the available commands.
echo            Echo the given input.
ping            Pings the core.
banner          Shows a banner with the given input.
cpuinfo         Prints information about the current core.
sleep           Sleeps for a given number of seconds.
uptime          Tell how long the system has been running.
axpy            Run the axpy benchmark.
axpy_vector     Run the axpy vector version benchmark.
epac@nikitas$ cpuinfo
Chip Frequency    = 1000 MHz
Core HartID       = 0
Cycle Count       = 216041194340
Instruction Count = 1664256219
epac@nikitas$ uptime
 up for 0 hours 03 minutes 46 seconds
epac@nikitas$ axpy 8192
Running AXPY Scalar with 8192 array elements
init time: 352338 cycles

axpy scalar reference time: 281334 cycles

done
Result ok !!!
epac@nikitas$ axpy_vector 8192
Running AXPY Vector with 8192 array elements
init time: 352341 cycles

axpy vector time: 9725 cycles

done
Result ok !!!
epac@nikitas$
```

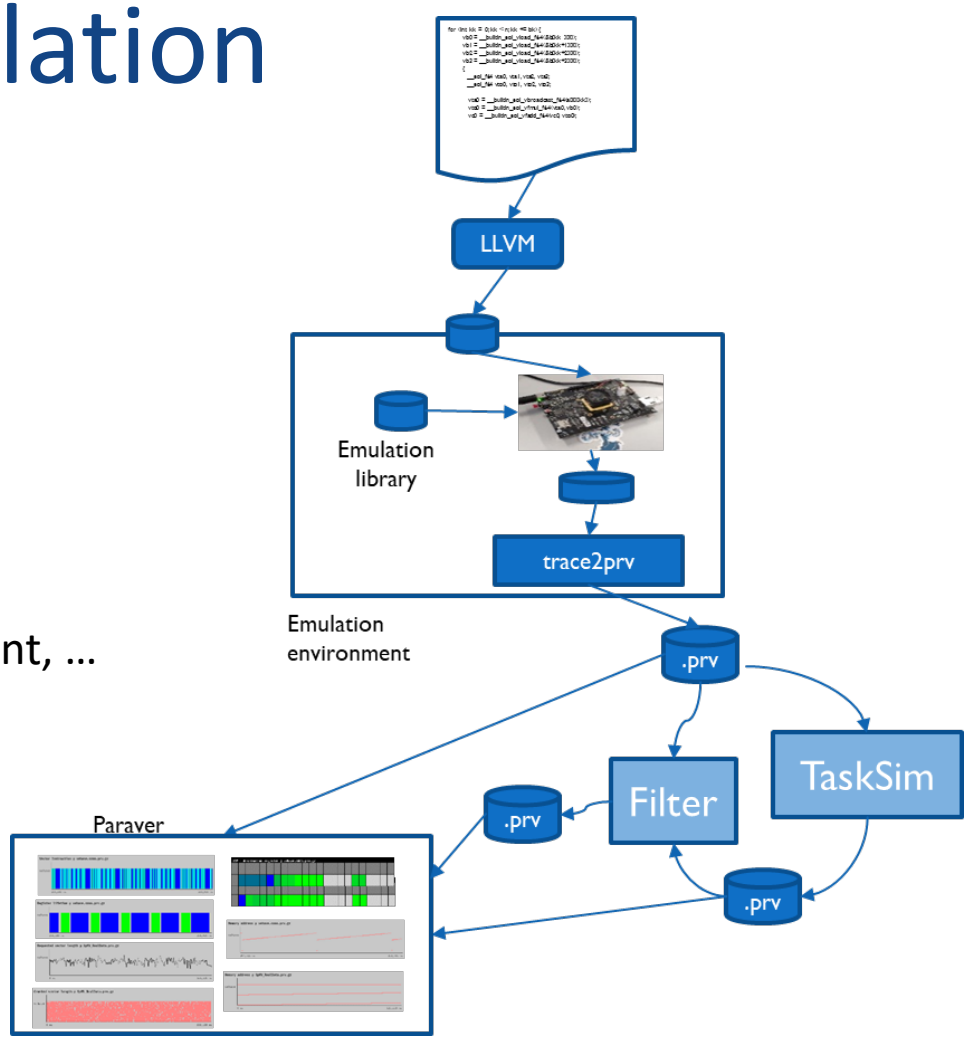
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

SDVs

RVV Software emulation

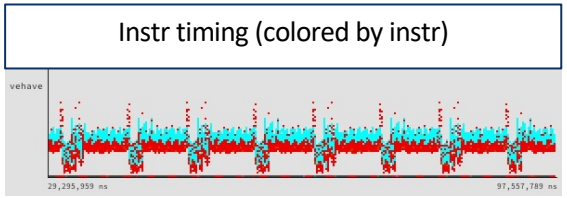
- Functional
  - Full OS, ….
  - On QEMU & scalar RISC-V cores
- Timing model
  - Microarchitectural parameters
    - MAXVL, OoO, Locality management, …

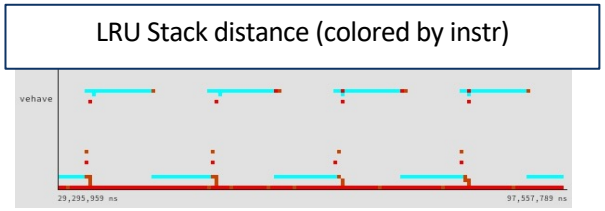


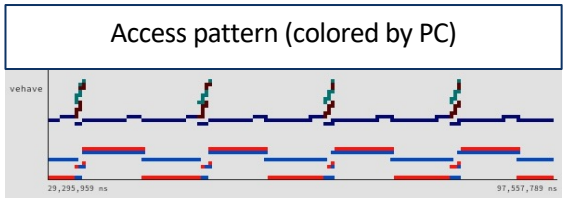
- https://repo.hca.bsc.es/gitlab/epi-public/risc-v-vector-simulation-environment

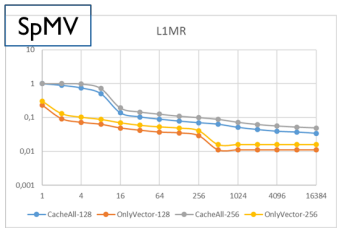
RVV Software emulation

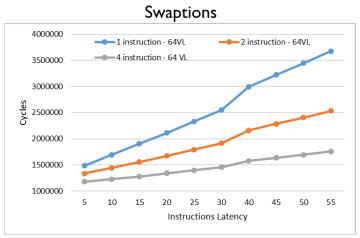
- Detailed analysis & insight



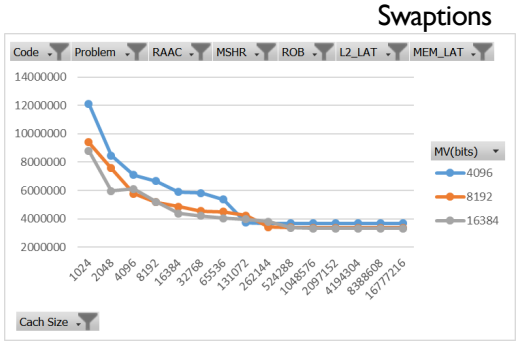


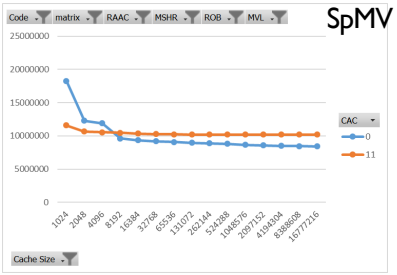






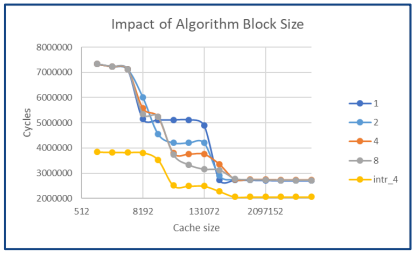
32KB



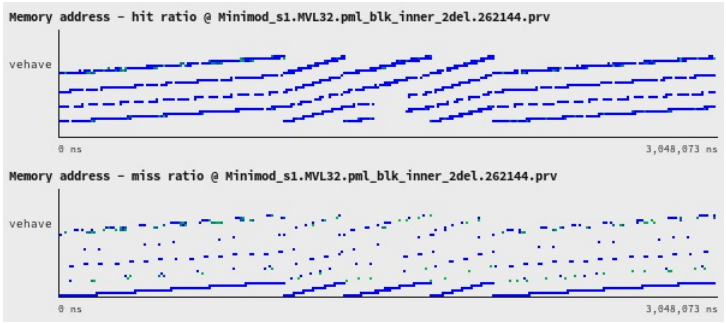


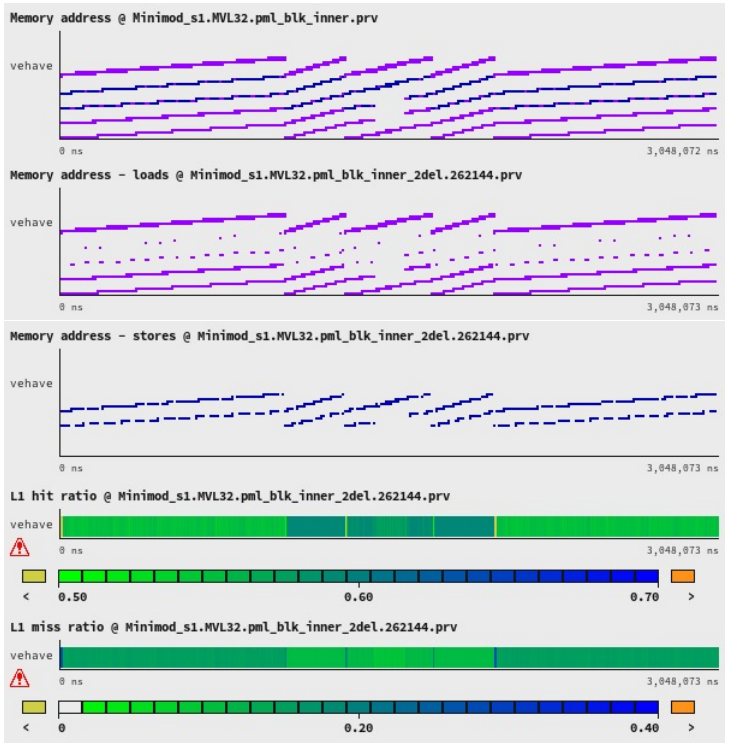
RVV Software emulation

- Detailed analysis & insight



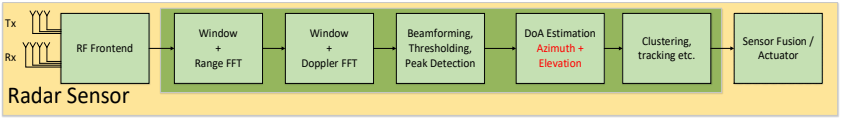
256K cache



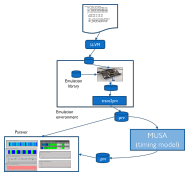


RVV Software emulation

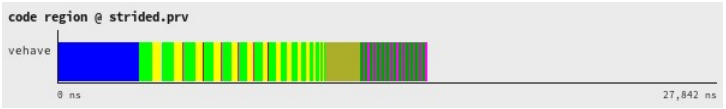
- Radar miniapp

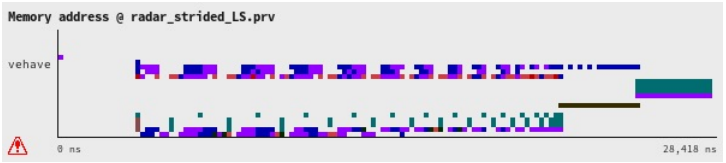


- Directives + automatic vectorization
  - Incremental way

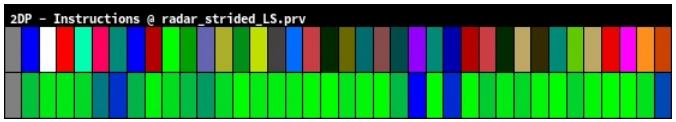


- Steering compiler optimizations
  - Complex data types
  - Indexed → strided
  - Reuse through registers
  - Avoid optimizations generating extremely short vector lengths





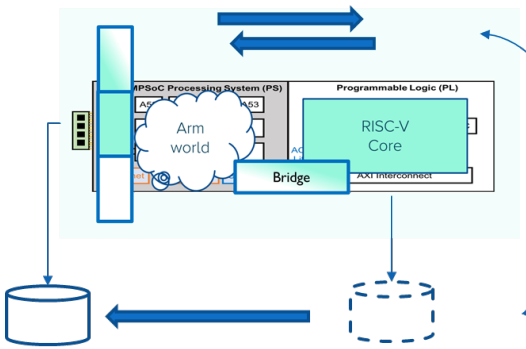
- Steering code refactoring
  - With productivity in mind !
    - Interchange, collapse, …



Heterogeneous ARM + RISC-V

- Linux Boot on 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



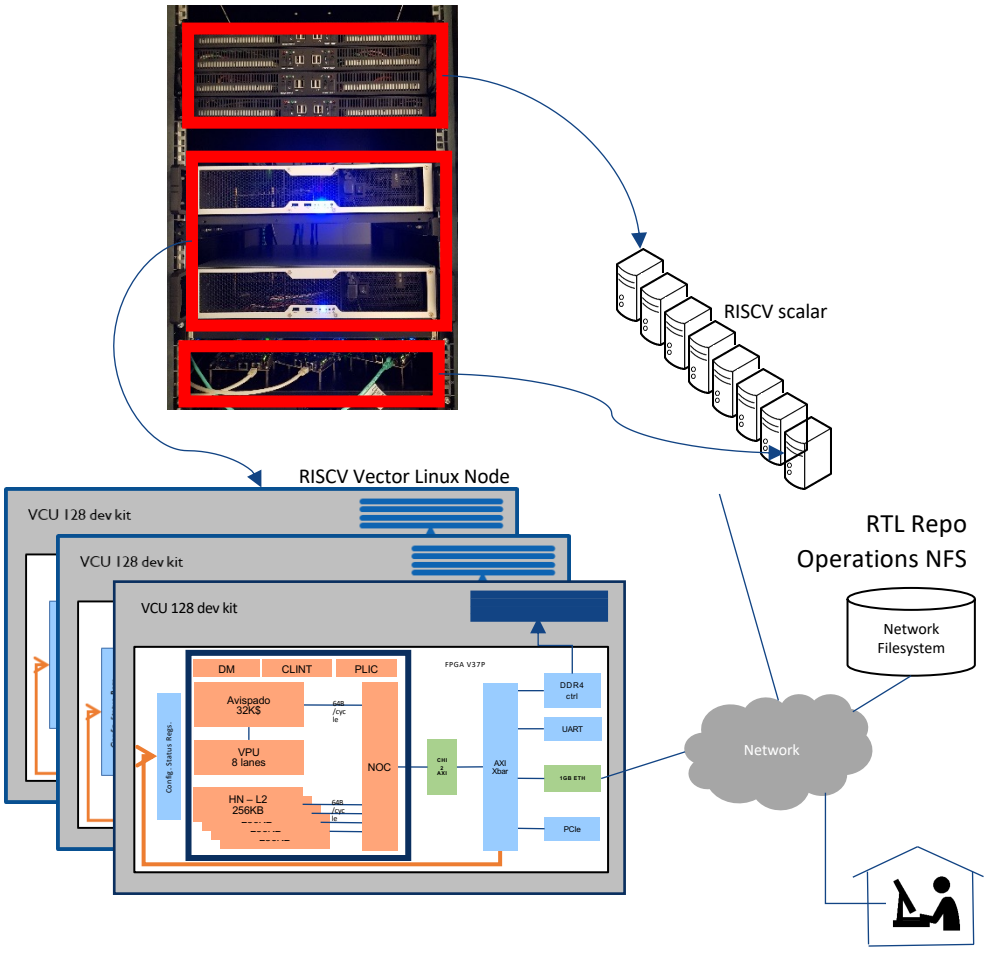

```
#on RISC-V side
$mkfs.ext4 -b 4096 /dev/vda
$mount -t ext4 /dev/vda /mnt/scratchfs
```

```
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
  - Test user codes @ real RTL
  - Give to EPI partners an external users early access to EPI technology
    - Two step procedure



RVV @ FPGA & ecosystem

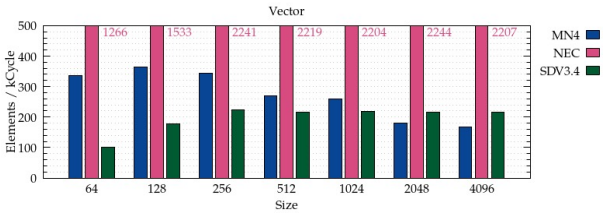
- Very preliminary but promising results
- Providing very useful feedback for EPAC1.5 co-design

```
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,
__epi_m1);
            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 = __bui
            v_x       = __bui
            v_prod    = __bui
            v_partial_res = _
v_partial_res, gvl);
            colid  += gvl;
        }
        y[row] += __builtin_e
    }
}
```

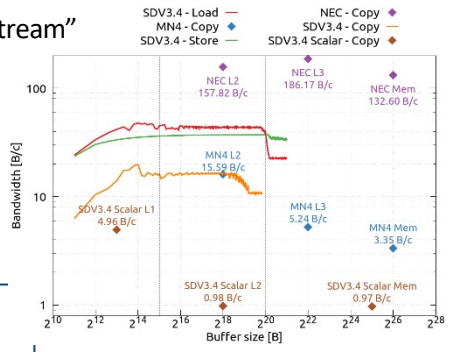
```
void target_inner_3d (…) {
  for (i) {
    for (j) {
      #pragma clang loop vectorize(enable)
      for (k) {
        lap = LAP;
        v[IDX3_l(i,j,k)] = 2.f*u[IDX3_l(i,j,k)]
                 -v[IDX3_l(i,j,k)]+vp[IDX3(i,j,k)]*lap;
      }}}}
```

```
…
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]);
}
```

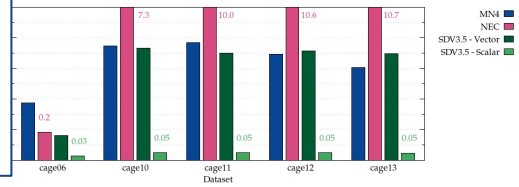
Jacobi 2D



"Stream"

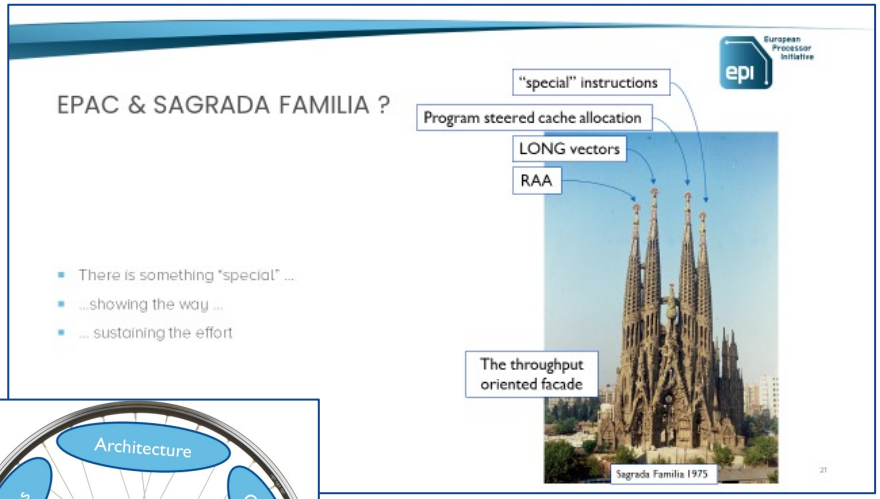


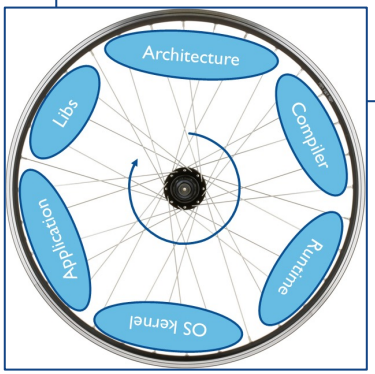
SpMV



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
- High bandwidth per control flow
- Push behaviour exploitation to low levels
- Co-ordination between levels
- Make it all look as classical sequential programming to ensure productivity





RISC-V

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Thanks