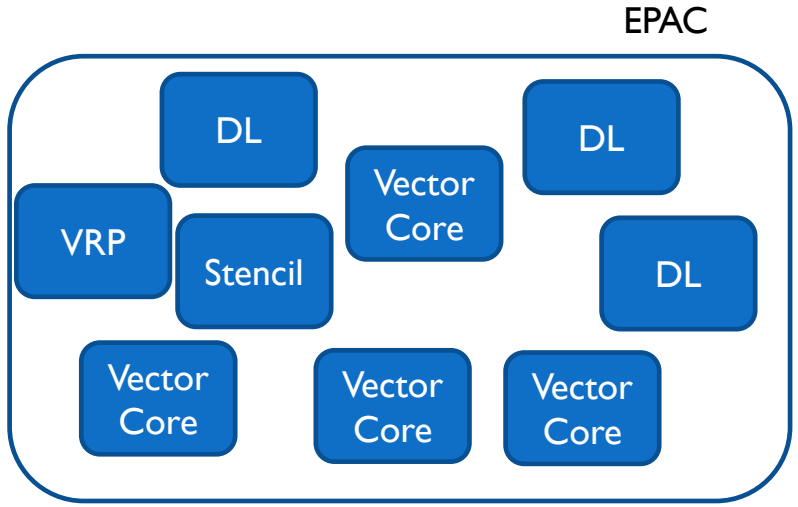
FUTURE HPC SYSTEMS MADE IN EUROPE

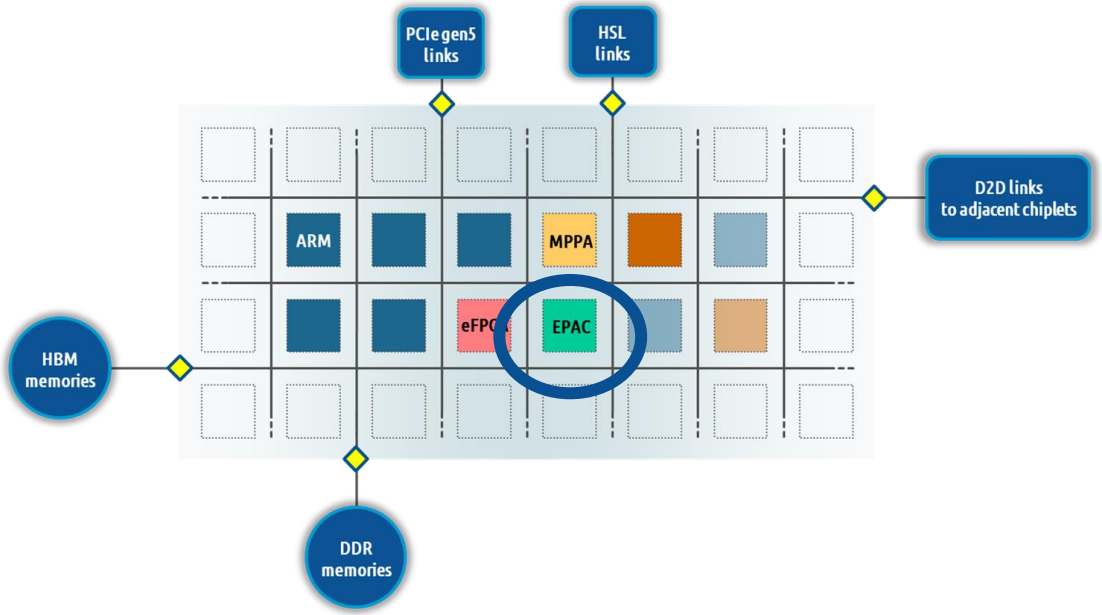
JESUS LABARTA (@BSC.ES)

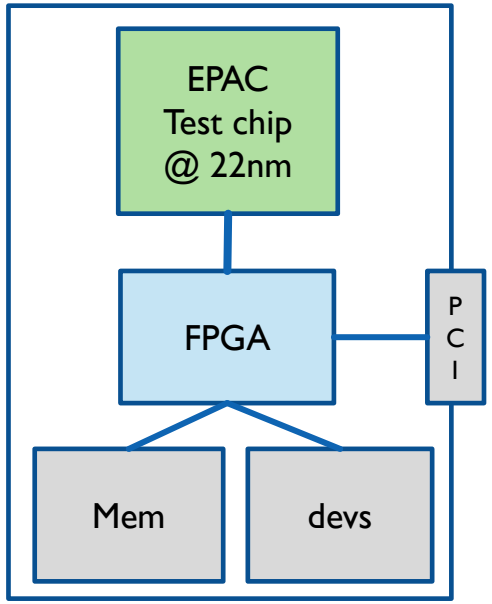
EPAC (EUROPEAN PROCESSOR ACCELERATOR)

- **Develop** and demonstrate **fully European** processor IPs based on the **RISC-V ISA**
  - Build on existing EU IP, leverage EU background and vision
  - BSC, CEA, Chalmers, ETH Zürich, Extoll, FH, FORTH, SMD, FhG, IST, Unibo, Unizg, E4
- Provide a very **low power** and high computing throughput **accelerator**
  - HPC
  - **Emerging** → Automotive
- Accelerator: “forgiving” entry point, towards general purpose HPC



EPAC OUTCOMES

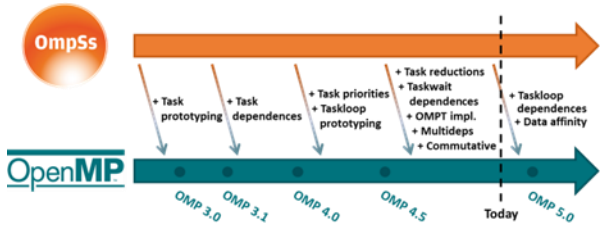




VISION CONTEXT

Insight on behavior

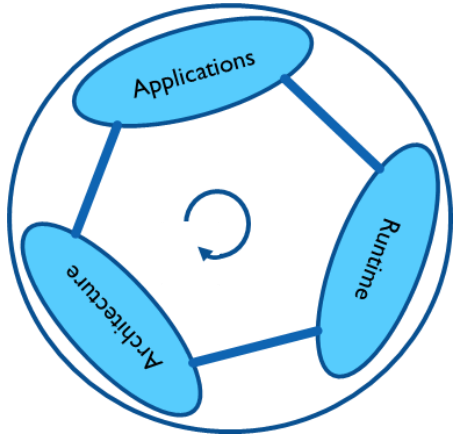
The programmer interface

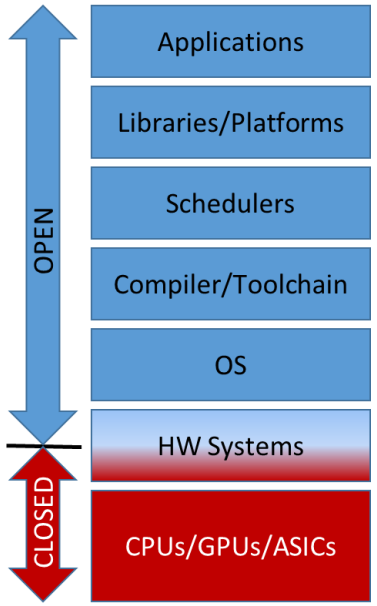


RISC-V

Leverage standards
Opportunity to innovate

Holistic Co-design

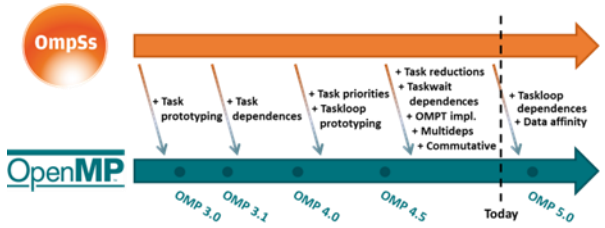




VISION CONTEXT

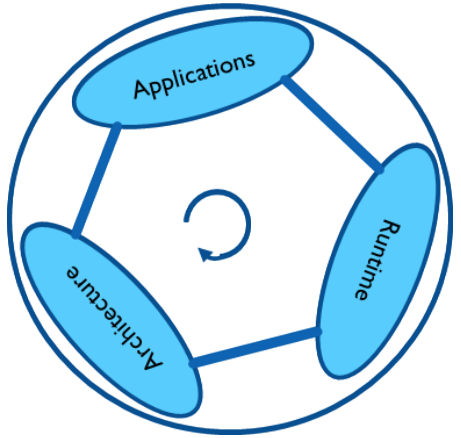
Insight on behavior

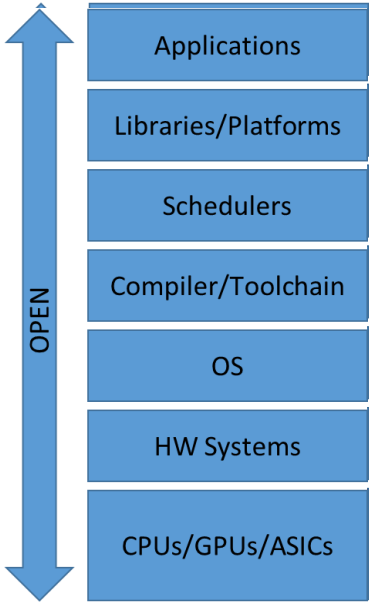
The programmer interface



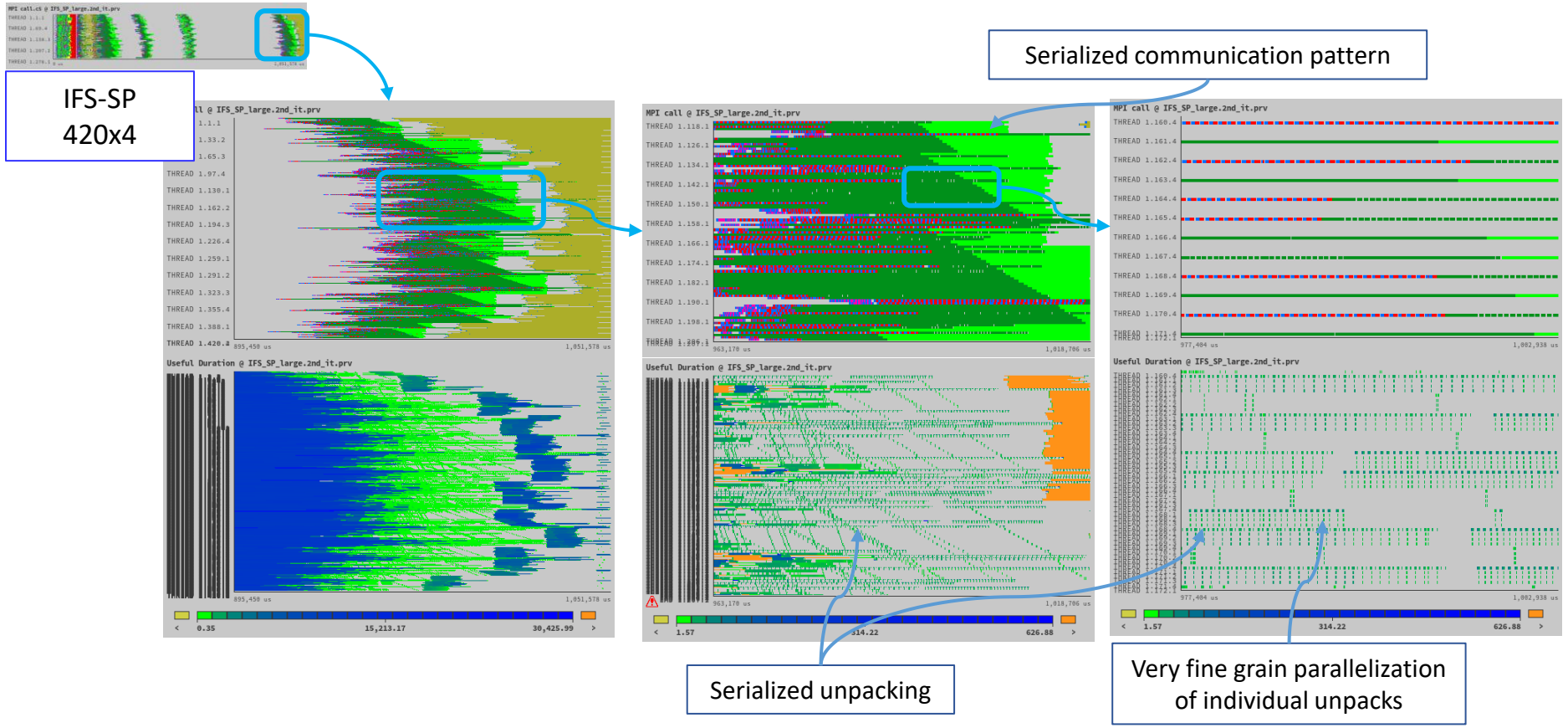
Leverage standards
Opportunity to innovate

Holistic Co-design



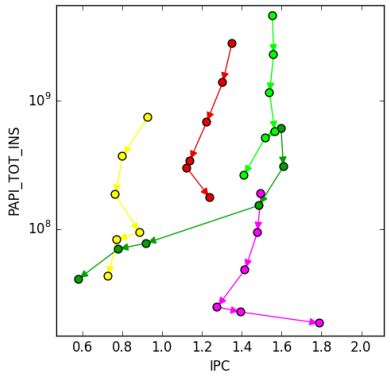


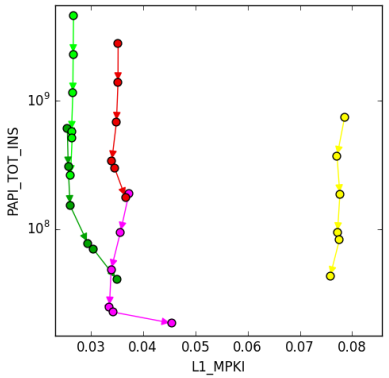
Insight on applications

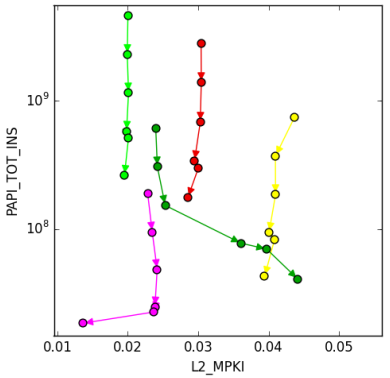


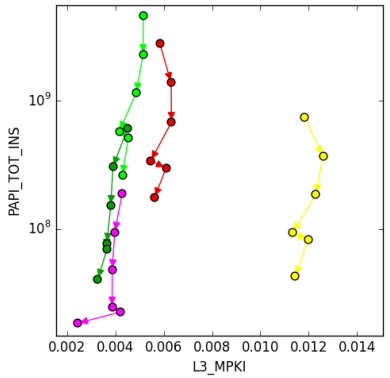
Insight on applications

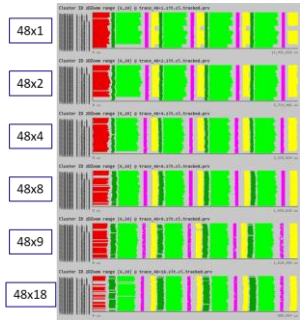
- Tracking IPC factors with scale

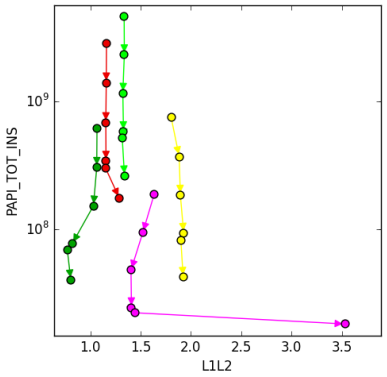


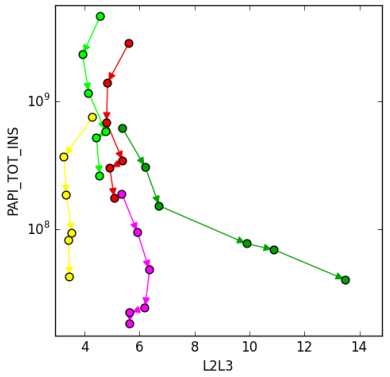






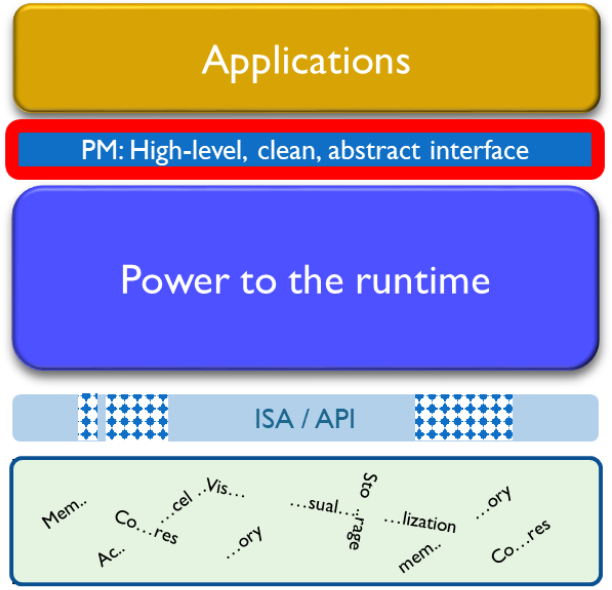






STEERING SOFTWARE …

- Best practices in parallel programming …
  - Hierarchy
    - Hybrid, Nesting, acceleration,
    - Homogenize Heterogeneity
  - Asynchrony
    - Task dependencies
    - Taskify communications
  - Malleability
  - Hinting through the osmotic membrane
    - Order: priority
    - Memory management & Locality
- … runtime development …
  - MPI + OpenMP
    - MPI InteroperabilityTask Aware MPI (TAMPI)
    - Dynamic Load balancing (DLB)



General purpose

Task & data based

Forget about resources

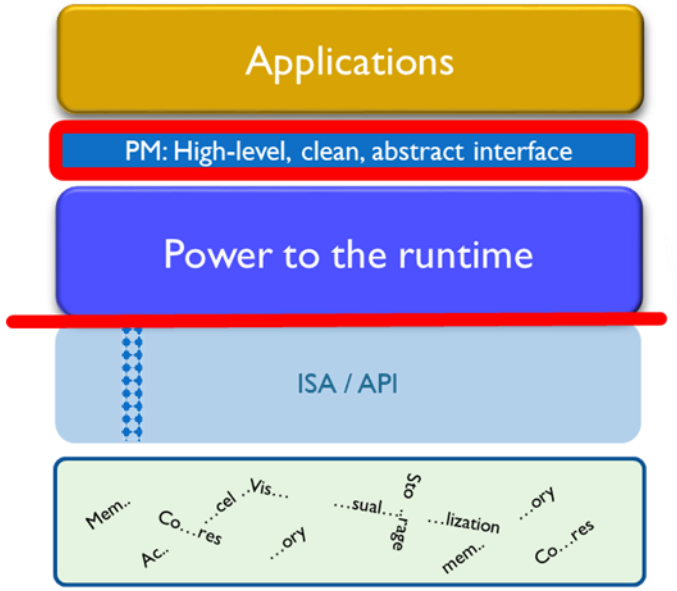
Decouple:
Minimal & sufficient permeability?

Intelligence
&
Resource management

"Reuse & expand" old architectural ideas under new constraints

… AND ARCHITECTURE

- Risc-V long vectors
  - Vector Length Agnostic : Dynamically adapt algorithmic structure to architecture
- Hierarchical balance between granularity levels (architecture and programming)
  - MPI – OpenMP (Nesting) - Long Vectors
  - “Limited” number of control flows
- Long vectors
  - Latency → throughput: decouple Front end – back end
  - Potential to optimize memory throughput: Convey access pattern semantics to the architecture, Locality hints …
  - High BW / control flow
- Leverage system software technologies (Operating system, Compiler)



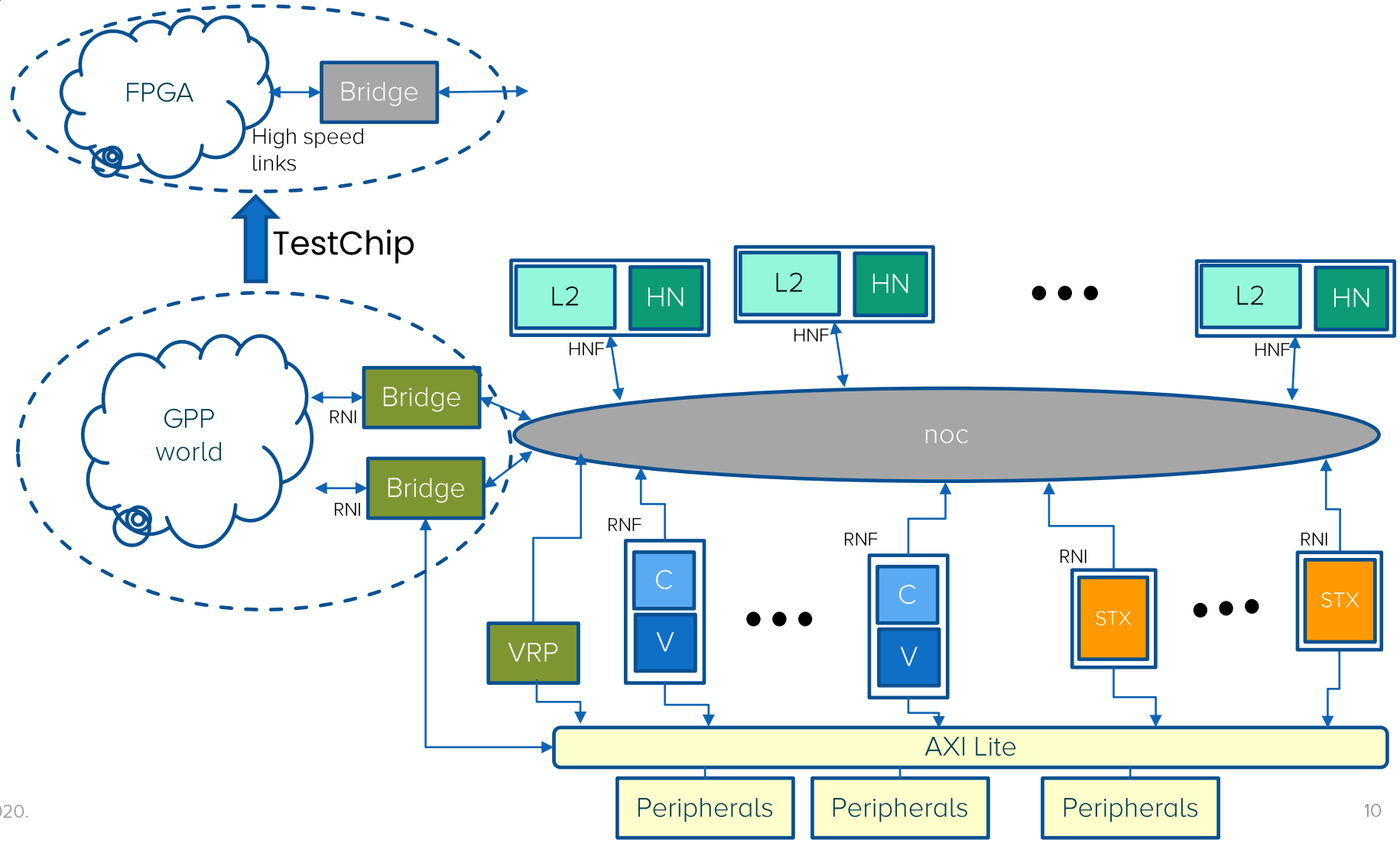
Vector ISA

Raise semantic level of interface

Possibility to reduce leakage

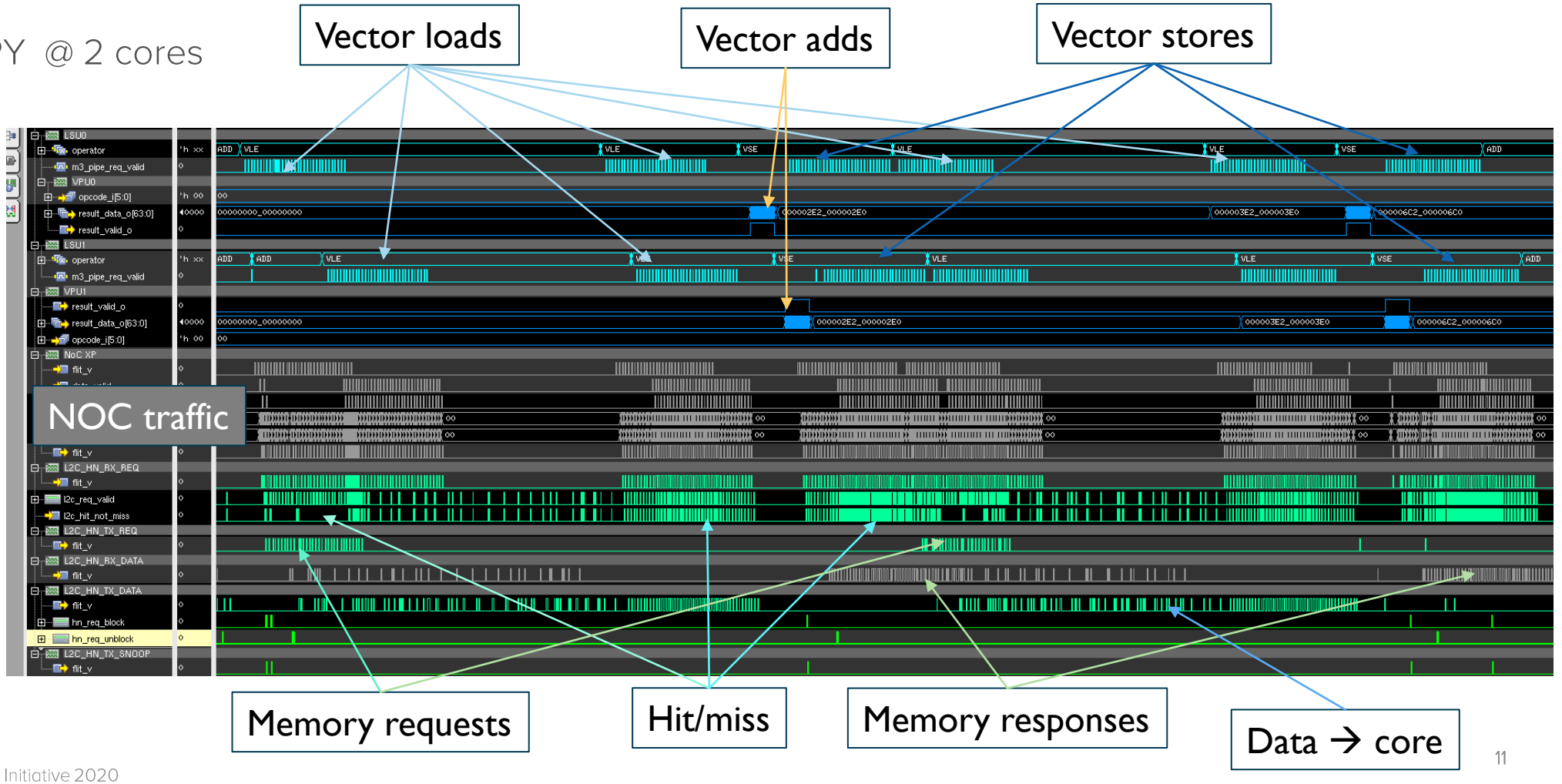
RTL “OWNERS”

- CEA
- Chalmers
- FORTH
- Semidynamics
- BSC
- UNIZG
- ETH + FhG
- EXTOLL
- ETH+ EXTOLL



INTEGRATION

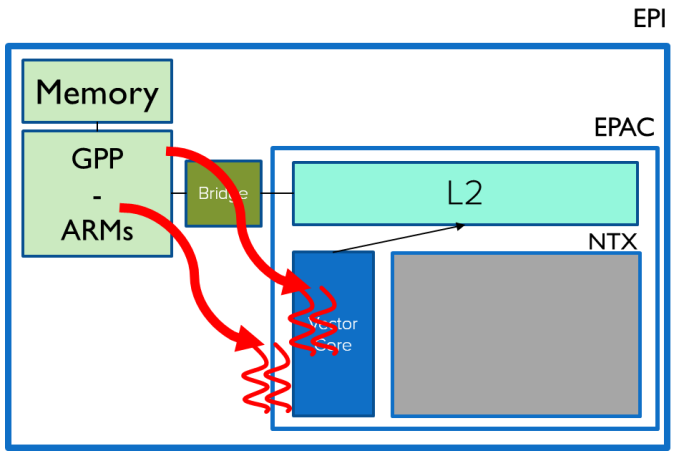
- Vector AXPY @ 2 cores



CEA
Chalmers
FORTH
Semidynamics
BSC
UNIZG
ETH + FhG
EXTOLL
ETH

PROGRAMMING MODEL

- MPI + OpenMP
  - Offloading
  - Tasks
  - SIMD
  - “Specific extensions”



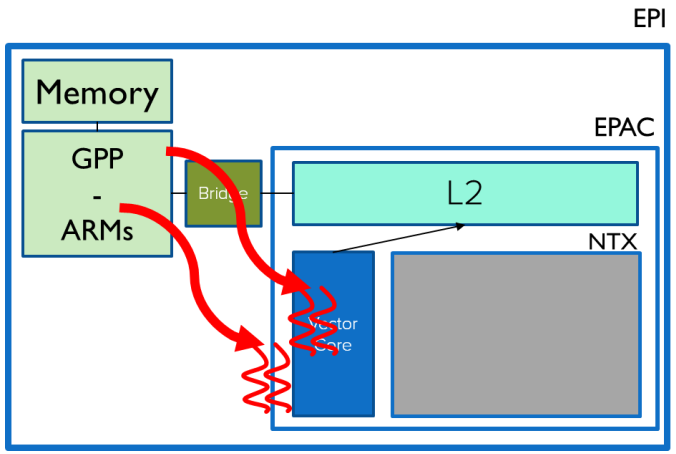

```
void axpy_omp_nest    (double a, double *dx, double *dy, int n) {
   int i, chunk;
   #pragma omp taskloop
   for (i=0; i<n; i+=TS) {
      chunk= n>i+TS? TS : n-i;
      #pragma omp target map(to:dx[i:i+chunk], tofrom:dy[i:i+chunk])
      axpy_omp        (a, &dx[i], &dy[i], chunk);
   }
```

```
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];
}
```

PROGRAMMING MODEL

- MPI + OpenMP
  - Offloading
  - Tasks
  - SIMD
  - “Specific extensions”



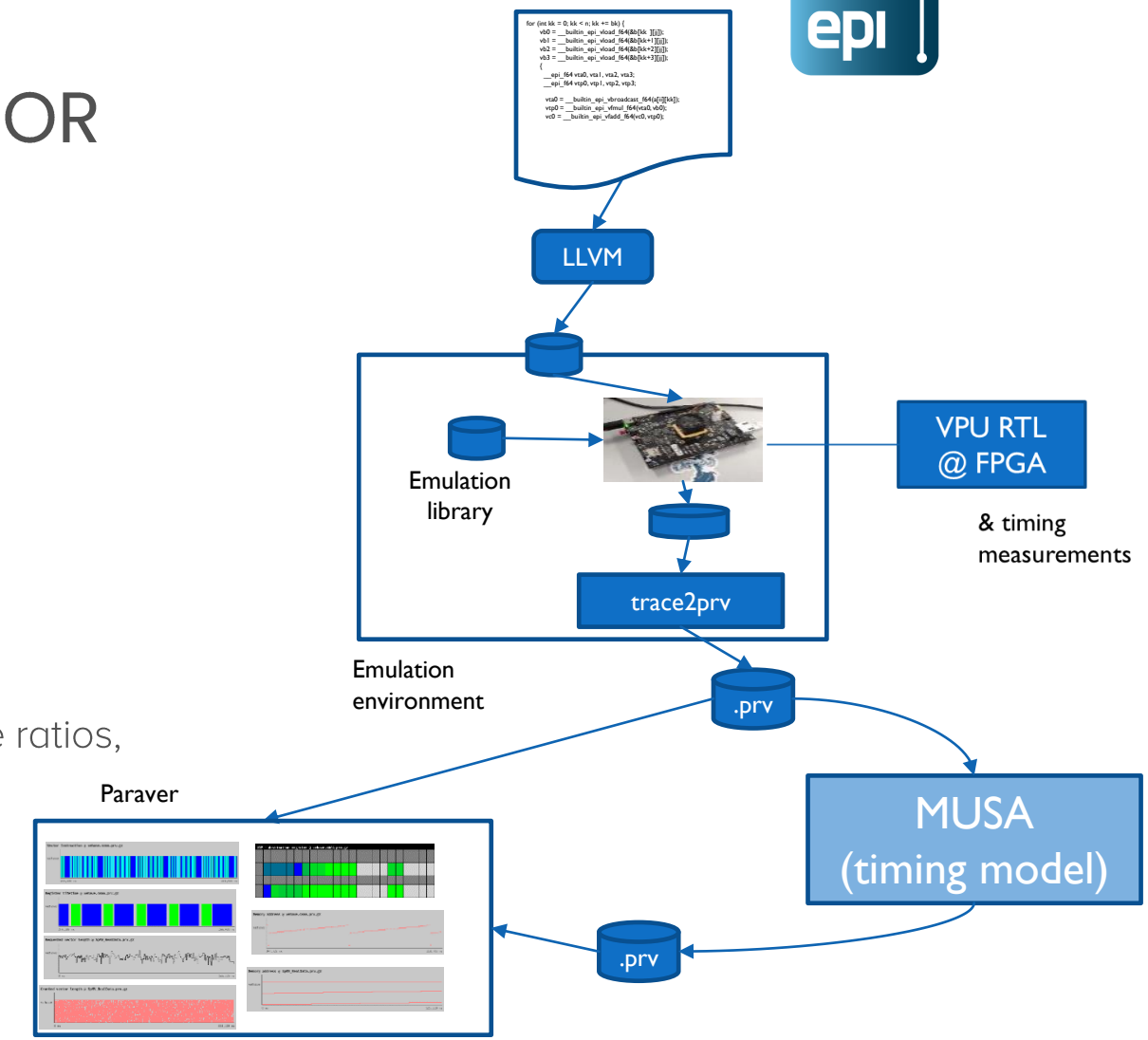

```
void axpy_intrinsics  (double a, double *dx, double *dy, int n) {
   int i;
   int gvl = __builtin_epi_vsetvl(n, __epi_e64, __epi_m1);
    __epi_1xf64 v_a = __builtin_epi_vbroadcast_1xf64(a, gvl);

   for (i=0; i<n; ) {
       gvl = __builtin_epi_vsetvl(n - i, __epi_e64, __epi_m1);
       __epi_1xf64 v_dx = __builtin_epi_vload_1xf64(&dx[i], gvl);
       __epi_1xf64 v_dy = __builtin_epi_vload_1xf64(&dy[i], gvl);
       __epi_1xf64 v_res = __builtin_epi_vfmacc_1xf64(v_dy, v_a, v_dx, gvl);
       __builtin_epi_vstore_1xf64(&dy[i], v_res, gvl);
       i += gvl;
    }
}
```

```
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];
}
```

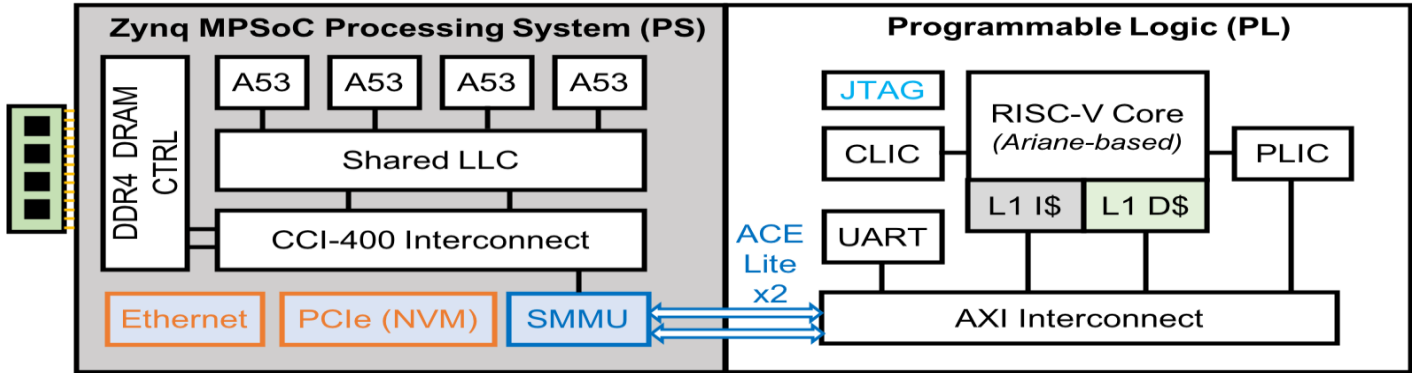
SDV1.2: RISC-V VECTOR EMULATOR

- Vectorizing compiler. LLVM.
- Emulation of RISC-V Vector ISA
  - Parametrized MAXVL
- Timing models
  - Memory architecture
- Very detailed analytics in Paraver
  - Vector length, register use, memory addresses, cache ratios, instruction timing
- Co-design
  - Kernels & Miniapps

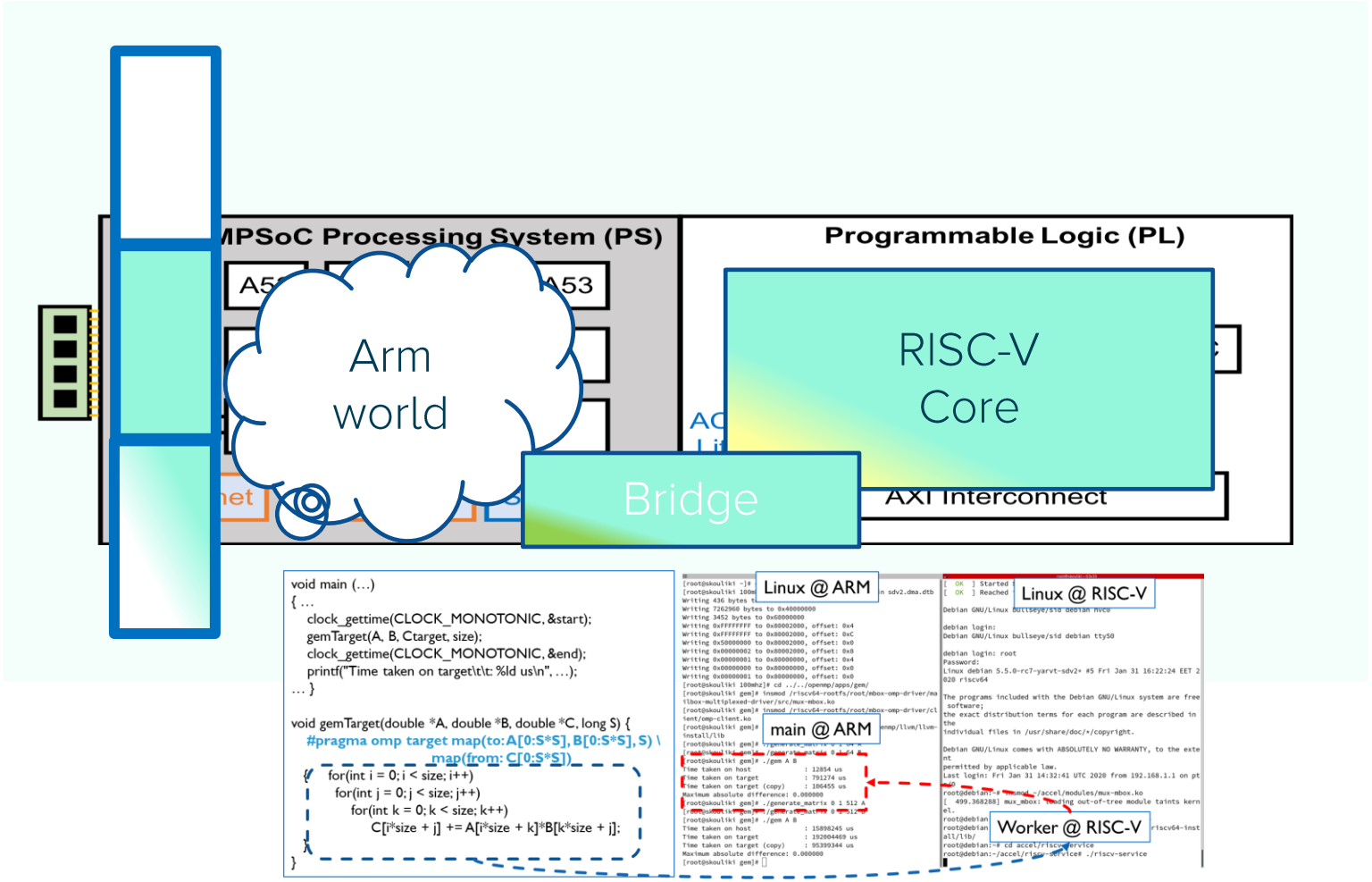


Available if you are interested in evaluating the framework and provide co-design input

SDV2.1: EPI HETEROGENEOUS PLATFORM (ARM + RISC-V)

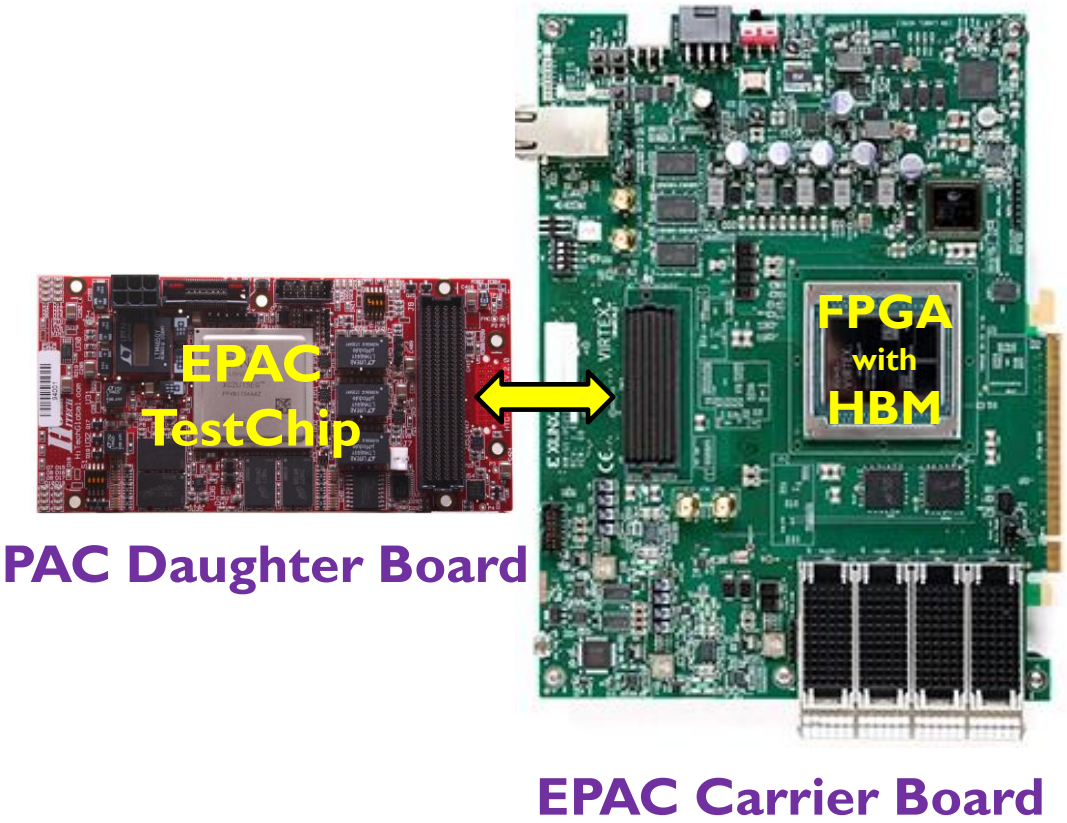


SDV2.1: EPI HETEROGENEOUS PLATFORM (ARM + RISC-V)



SDV / TEST BOARD

- SDV3
  - High end FPGA implementation
  - Standalone EPAC
  - HBM
  - Software development.
  - Flexible architectural early extension
- Test board for EPAC Test chip
- Integration in automotive MCP demonstrator



EPAC

- 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

- Contact us if you are interested in evaluating the framework and provide co-design input