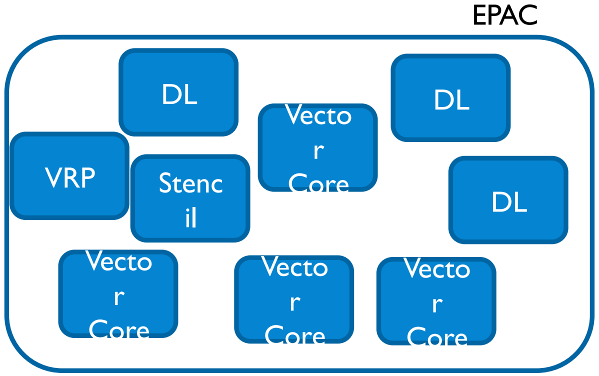
European
Processor
Initiative

FUTURE HPC SYSTEMS MADE IN EUROPE

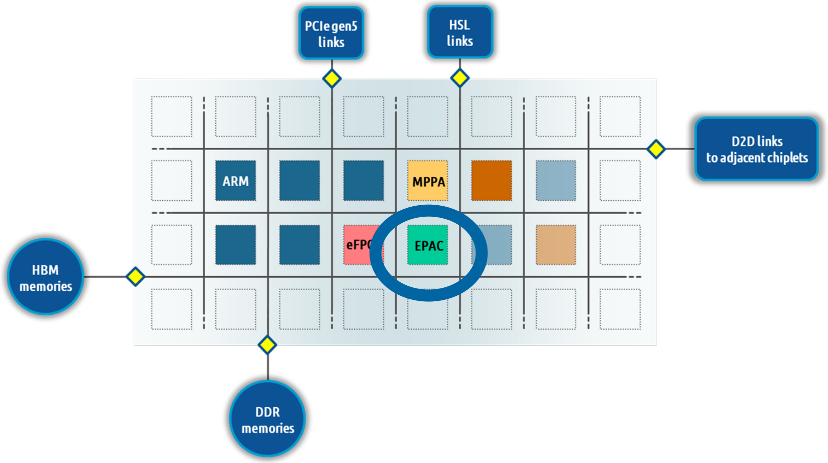
JESUS LABARTA (@BSC.ES)

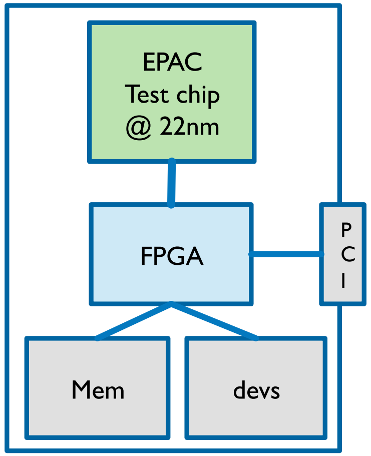
EPAC (EUROPEAN PROCESSOR ACCELERATOR)

- **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

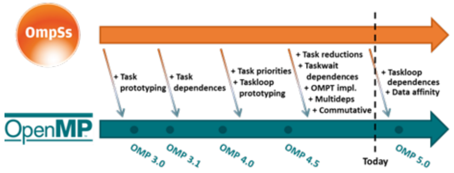




VISION CONTEXT

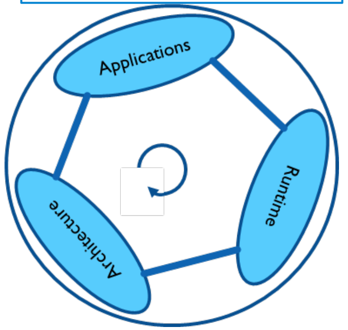
Insight on behavior

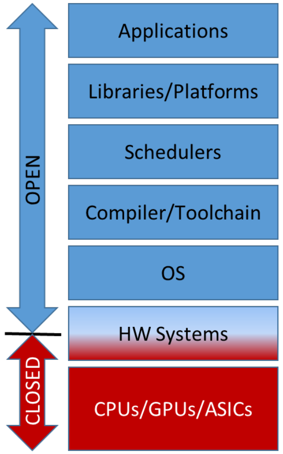
The programmer interface



Leverage standards
Opportunity to innovate

Holistic Co-design



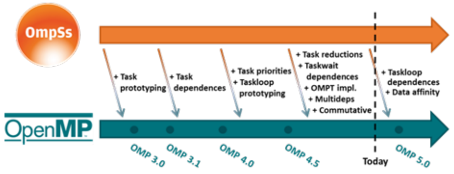


Applications
Libraries/Platforms
Schedulers
Compiler/Toolchain
OS
HW Systems
CPUs/GPUs/ASICs

VISION CONTEXT

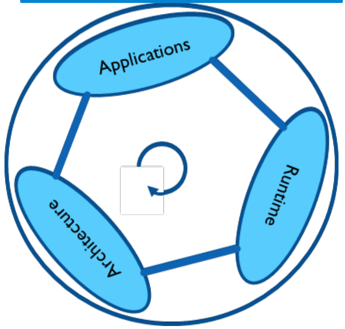
Insight on behavior

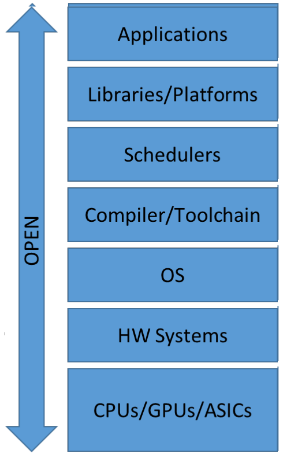
The programmer interface



Leverage standards
Opportunity to innovate

Holistic Co-design





Applications

Libraries/Platforms

Schedulers

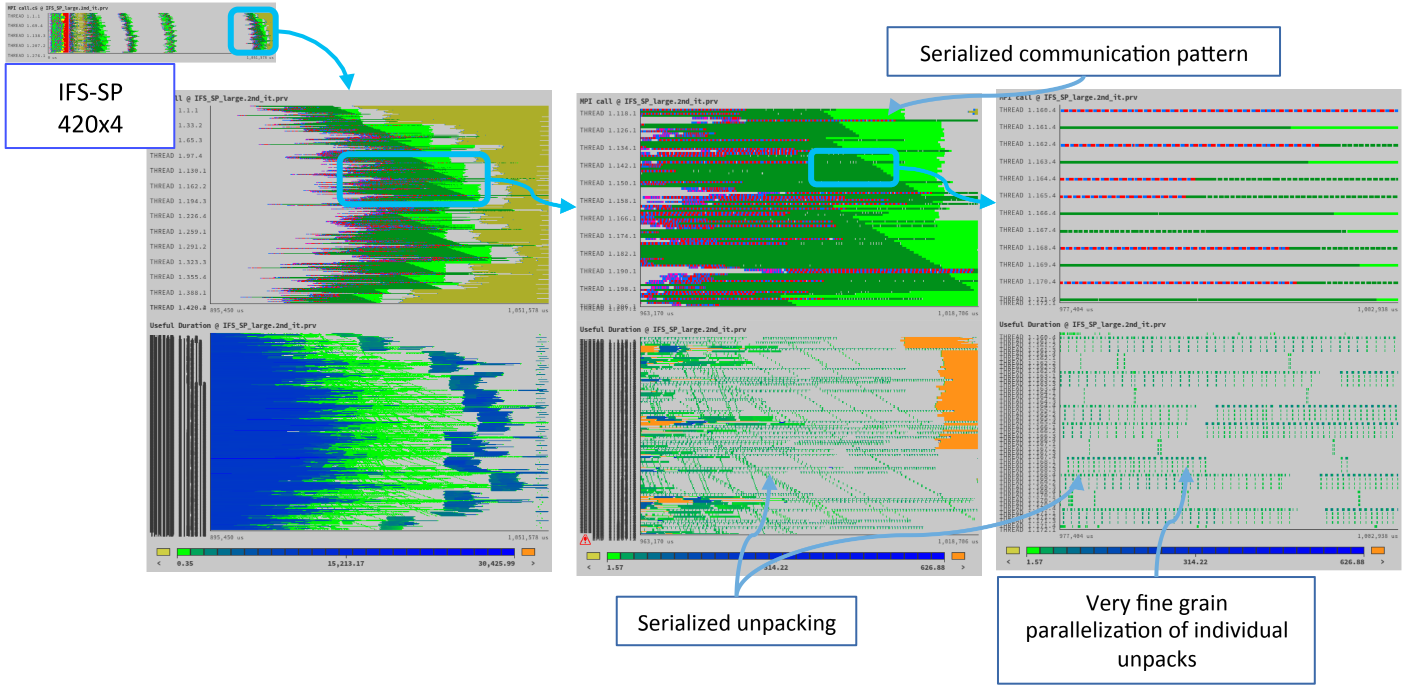
Compiler/Toolchain

OS

HW Systems

CPUs/GPUs/ASICs

Insight on applications


MPI call @ IFS_SP_large.2nd_it.prv
Useful Duration @ IFS_SP_large.2nd_it.prv

IFS-SP
420x4

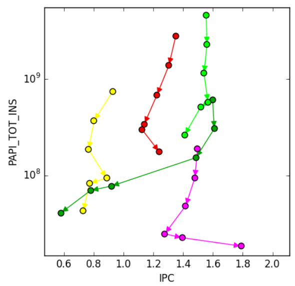
Serialized communication pattern

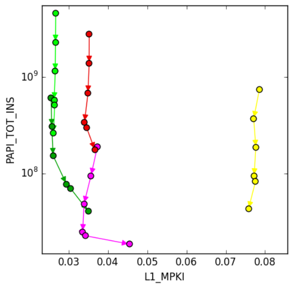
Serialized unpacking

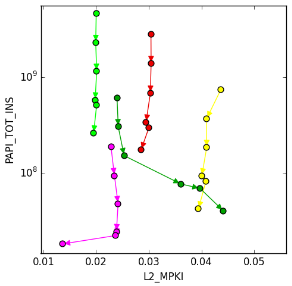
Very fine grain parallelization of individual unpacks

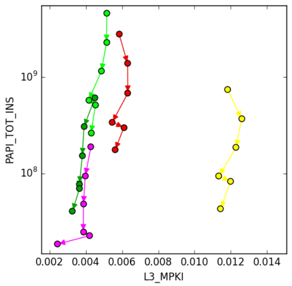
Insight on applications

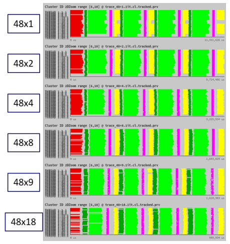
- Tracking IPC factors with scale

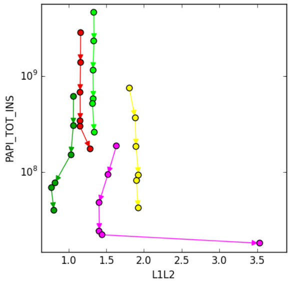


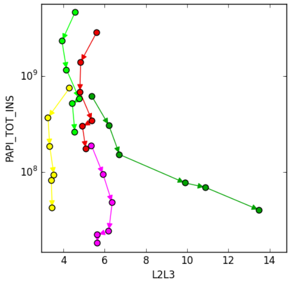






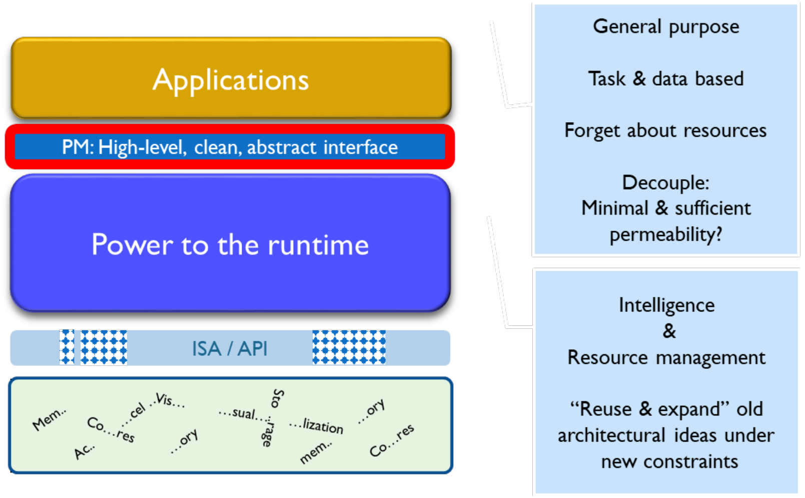






STEERING SOFTWARE …

- 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)



Applications

PM: High-level, clean, abstract interface

Power to the runtime

ISA / API

General purpose

Task & data based

Forget about resources

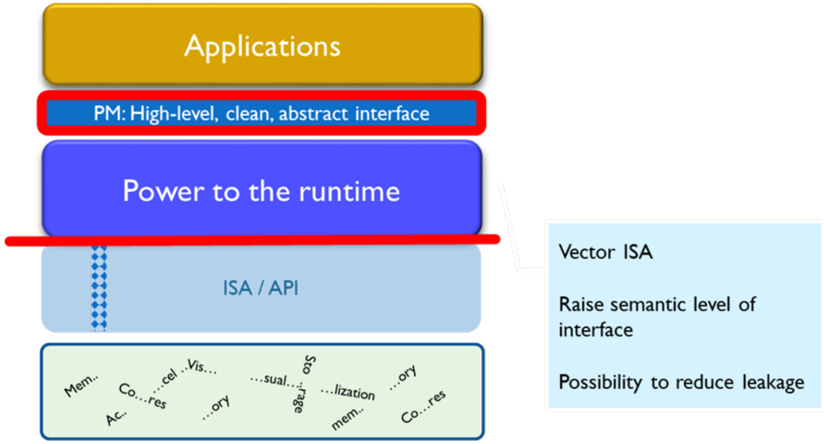
Decouple:
Minimal & sufficient
permeability?

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)



Applications

PM: High-level, clean, abstract interface

Power to the runtime

ISA / API

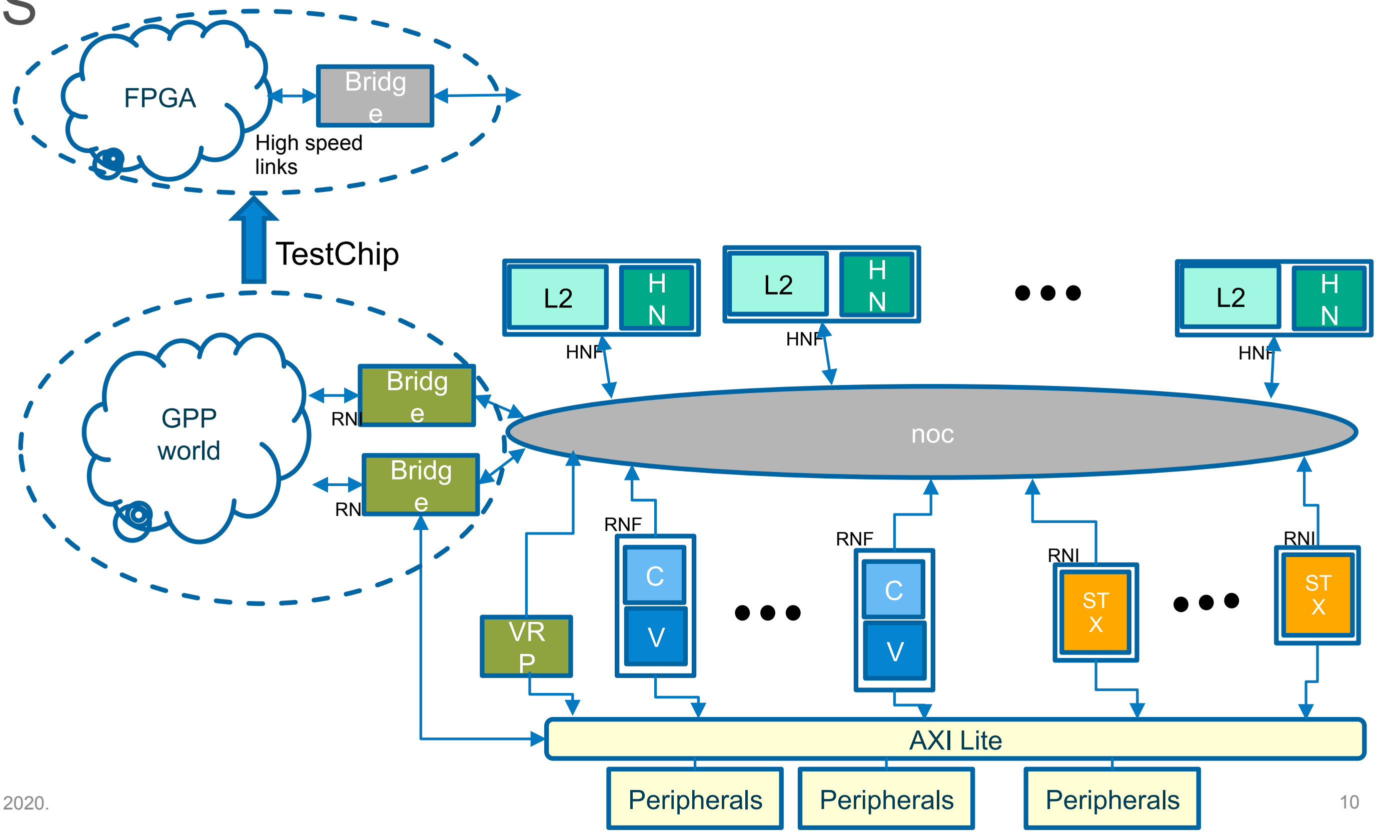
Vector ISA

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

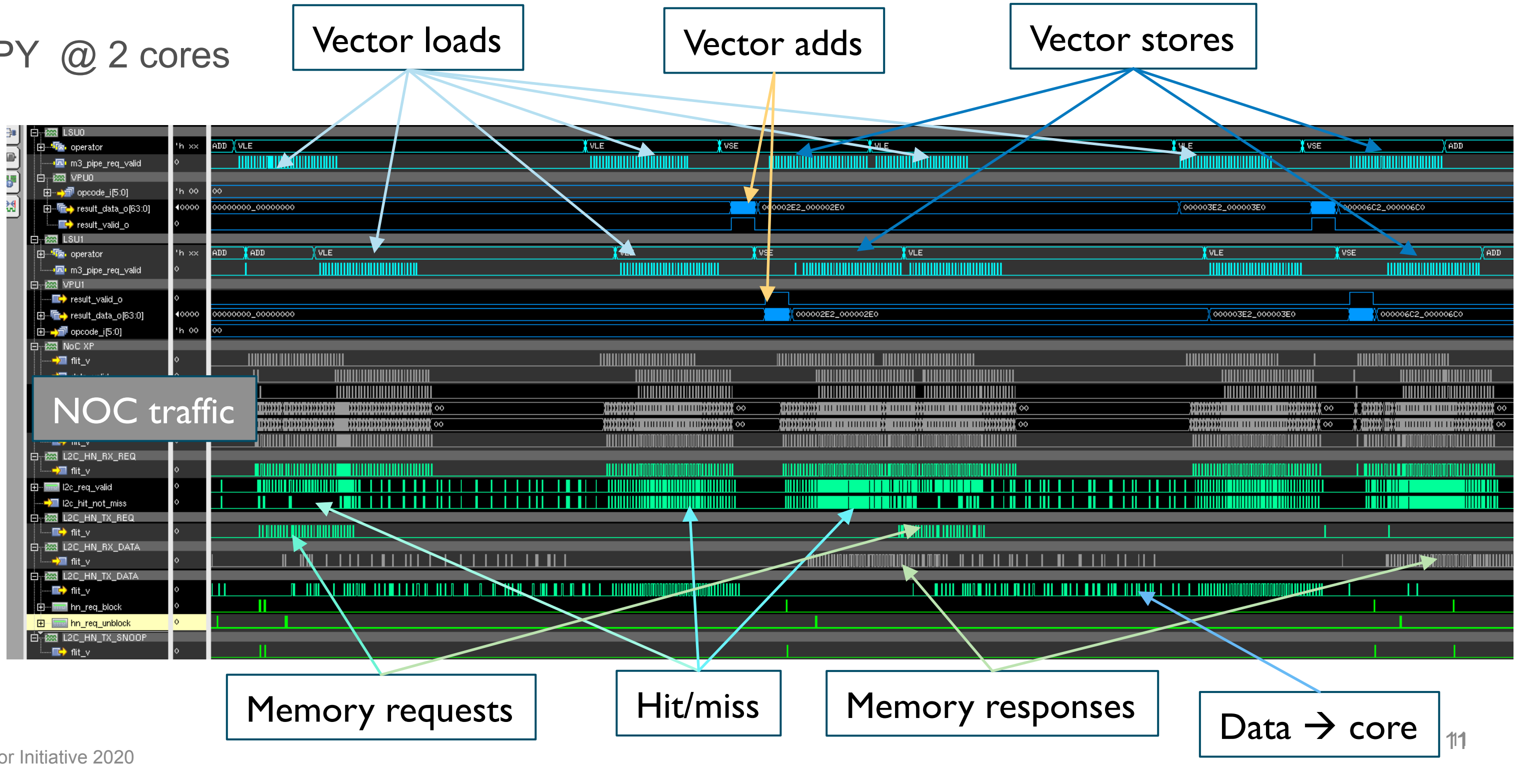
BSC

UNIZG

ETH + FhG

EXTOLL

ETH



Vector loads

Vector adds

Vector stores

NOC traffic

Memory requests

Hit/miss

Memory responses

Data → core

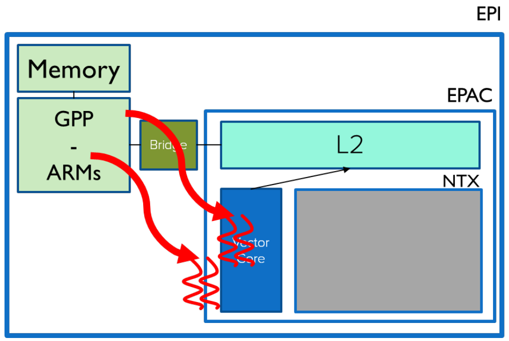
PROGRAMMING MODEL

- 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);
   }
```

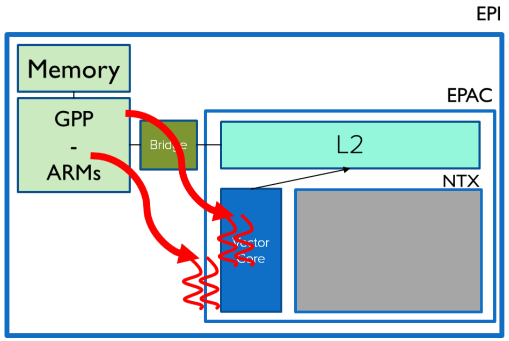
```
void axpy_omp         (double a, double *dx, double *dy, int n) {
   int I, chunk;
   #pragma omp taskloop
   for (i=0; i<n; i+=TS) {
      chunk= n>i+TS? TS : n-i;
      axpy_SIMD       (a, &dx[i], &dy[i], chunk);
   }
}
```

```
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”

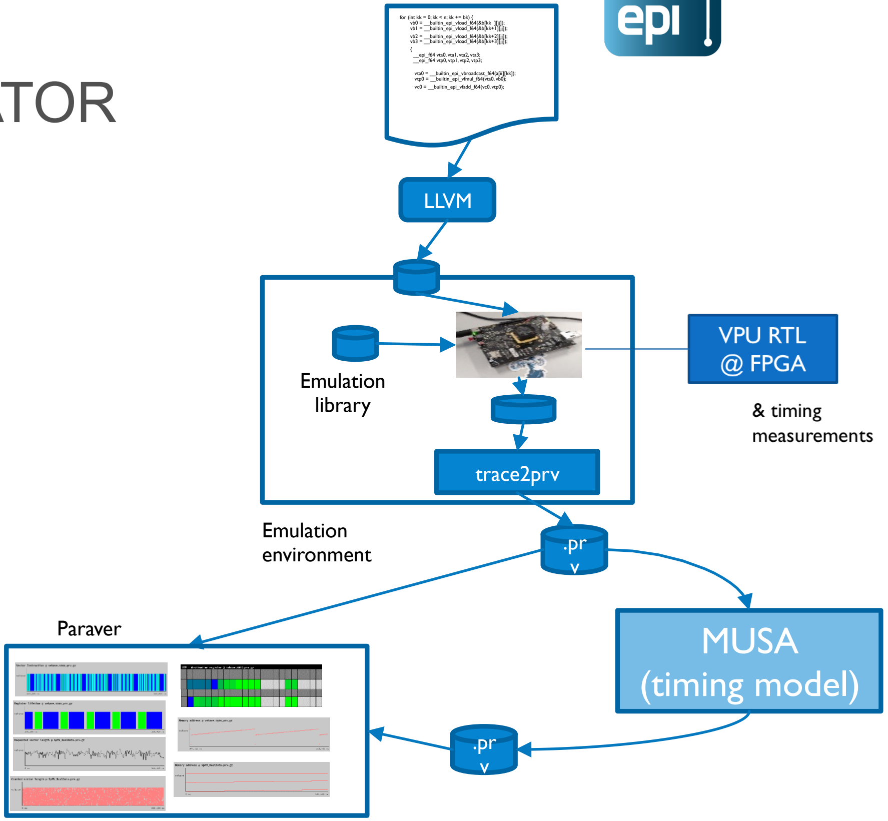


```
void axpy_intrinsics  (double a, double *dx, double *dy, int n) {
   int i;
   int gvl = __builtin_epi_vsetvl(n, __epi_e64, __epi_m1);
    __epi_1xf64 v_a = __builtin_epi_vbroadcast_1xf64(a, gvl);

   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

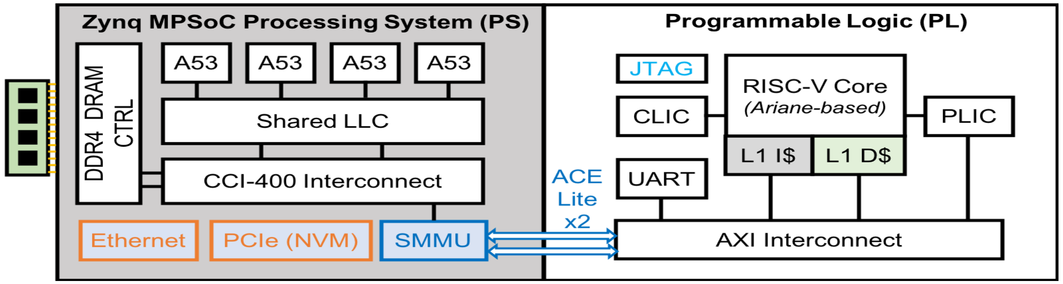
- Vectorizing compiler. LLVM.
- Emulation of RISC-V Vector ISA
  - **Parametrized MAXVL**
- Timing models
  - Memory architecture
- Very detailed analytics in Paraver
  - Vector length, register use, memory addresses, cache ratios, instruction timing
- Co-design
  - Kernels & Miniapps



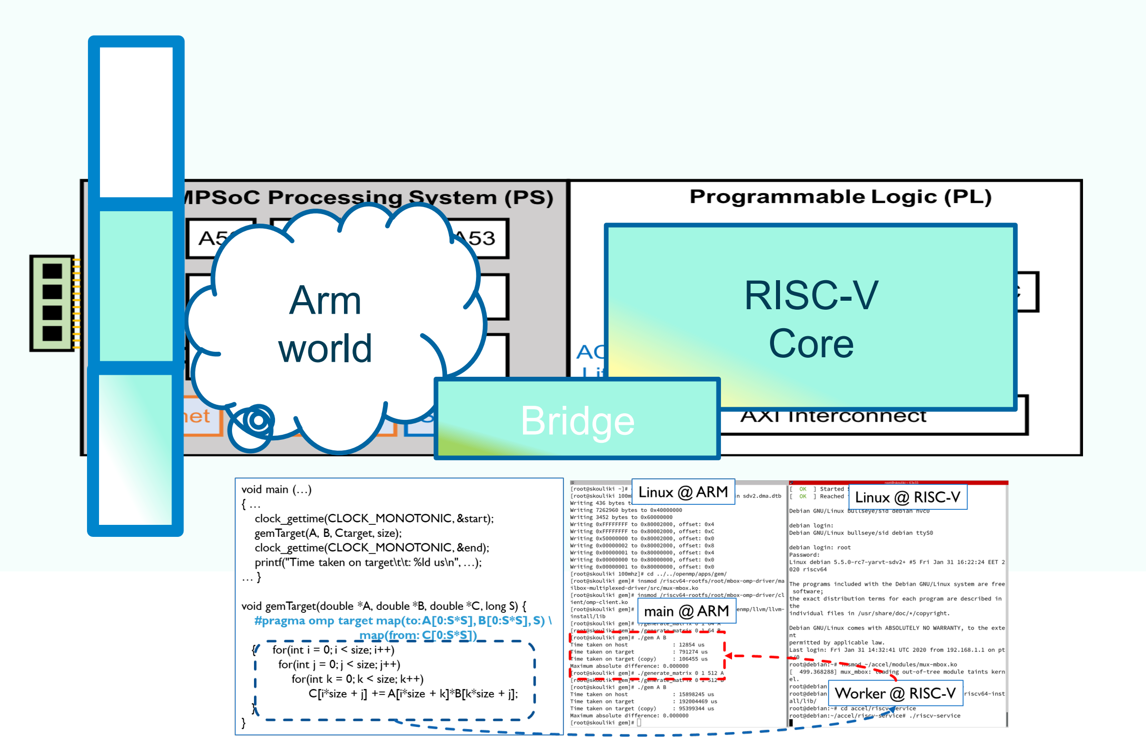
```
for (int kk = 0; kk < n; kk += bk) {
      vb0 = __builtin_epi_vload_f64(&b[kk  ][jj]);
      vb1 = __builtin_epi_vload_f64(&b[kk+1][jj]);
      vb2 = __builtin_epi_vload_f64(&b[kk+2][jj]);
      vb3 = __builtin_epi_vload_f64(&b[kk+3][jj]);
      {
        __epi_f64 vta0, vta1, vta2, vta3;
        __epi_f64 vtp0, vtp1, vtp2, vtp3;

         vta0 = __builtin_epi_vbroadcast_f64(a[ii][kk]);
         vtp0 = __builtin_epi_vfmul_f64(vta0, vb0);
         vc0 = __builtin_epi_vfadd_f64(vc0, vtp0);
```

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)



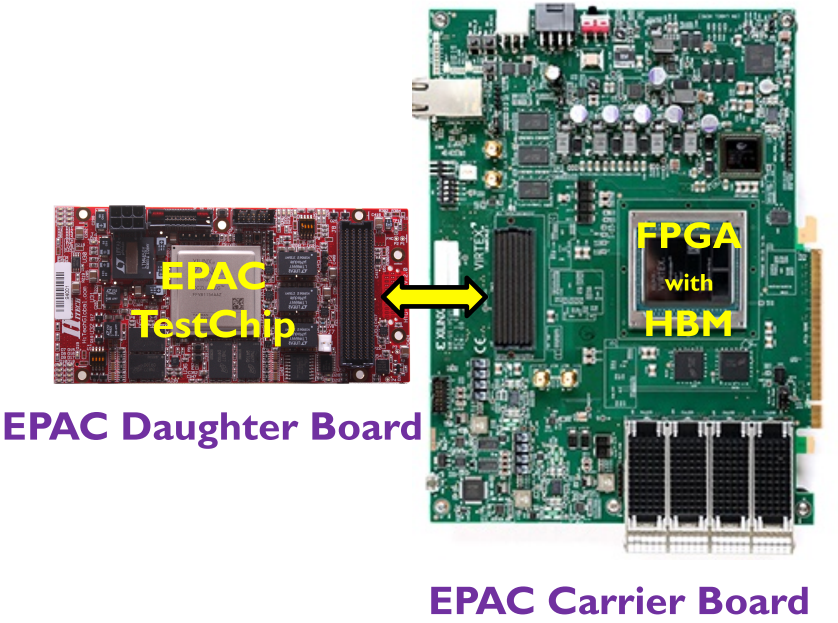
```
void main (…)
{ …
    clock_gettime(CLOCK_MONOTONIC, &start);
    gemTarget(A, B, Ctarget, size);
    clock_gettime(CLOCK_MONOTONIC, &end);
    printf("Time taken on target\t\t: %ld us\n", …);
… }

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];
   }
}
```

```
[root@skouliki ~]#
[root@skouliki 100m                                          n sdv2.dma.dtb
Writing 436 bytes t
Writing 7262960 bytes to 0x40000000
Writing 3452 bytes to 0x60000000
Writing 0xFFFFFFFF to 0x80002000, offset: 0x4
Writing 0xFFFFFFFF to 0x80002000, offset: 0xC
Writing 0x50000000 to 0x80002000, offset: 0x0
Writing 0x00000002 to 0x80002000, offset: 0x8
Writing 0x00000001 to 0x80000000, offset: 0x4
Writing 0x00000000 to 0x80000000, offset: 0x0
Writing 0x00000001 to 0x80000000, offset: 0x0
[root@skouliki 100mhz]# cd ../../openmp/apps/gem/
[root@skouliki gem]# insmod /riscv64-rootfs/root/mbox-omp-driver/ma
ilbox-multiplexed-driver/src/mux-mbox.ko
[root@skouliki gem]# insmod /riscv64-rootfs/root/mbox-omp-driver/cl
ient/omp-client.ko
[root@skouliki gem]#                                    enmp/llvm/llvm-
install/lib
[root@skouliki gem]# ./generate_matrix 0 1 64 A
[root@skouliki gem]# ./generate_matrix 0 1 64 B
[root@skouliki gem]# ./gem A B
Time taken on host              : 12854 us
Time taken on target            : 791274 us
Time taken on target (copy)     : 106455 us
Maximum absolute difference: 0.000000
[root@skouliki gem]# ./generate_matrix 0 1 512 A
[root@skouliki gem]# ./generate_matrix 0 1 512 B
[root@skouliki gem]# ./gem A B
Time taken on host              : 15898245 us
Time taken on target            : 192004469 us
Time taken on target (copy)     : 95399344 us
Maximum absolute difference: 0.000000
[root@skouliki gem]#
```

```
[  OK  ] Started
[  OK  ] Reached
Debian GNU/Linux bullseye/sid debian hvc0

debian login:
Debian GNU/Linux bullseye/sid debian ttyS0

debian login: root
Password:
Linux debian 5.5.0-rc7-yarvt-sdv2+ #5 Fri Jan 31 16:22:24 EET 2
020 riscv64

The programs included with the Debian GNU/Linux system are free
 software;
the exact distribution terms for each program are described in
the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the exte
nt
permitted by applicable law.
Last login: Fri Jan 31 14:32:41 UTC 2020 from 192.168.1.1 on pt
s/0
root@debian:~# insmod ~/accel/modules/mux-mbox.ko
[  499.368288] mux_mbox: loading out-of-tree module taints kern
el.
root@debian
root@debian                                             riscv64-inst
all/lib/
root@debian:~# cd accel/riscv-service
root@debian:~/accel/riscv-service# ./riscv-service
```

SDV / TEST BOARD

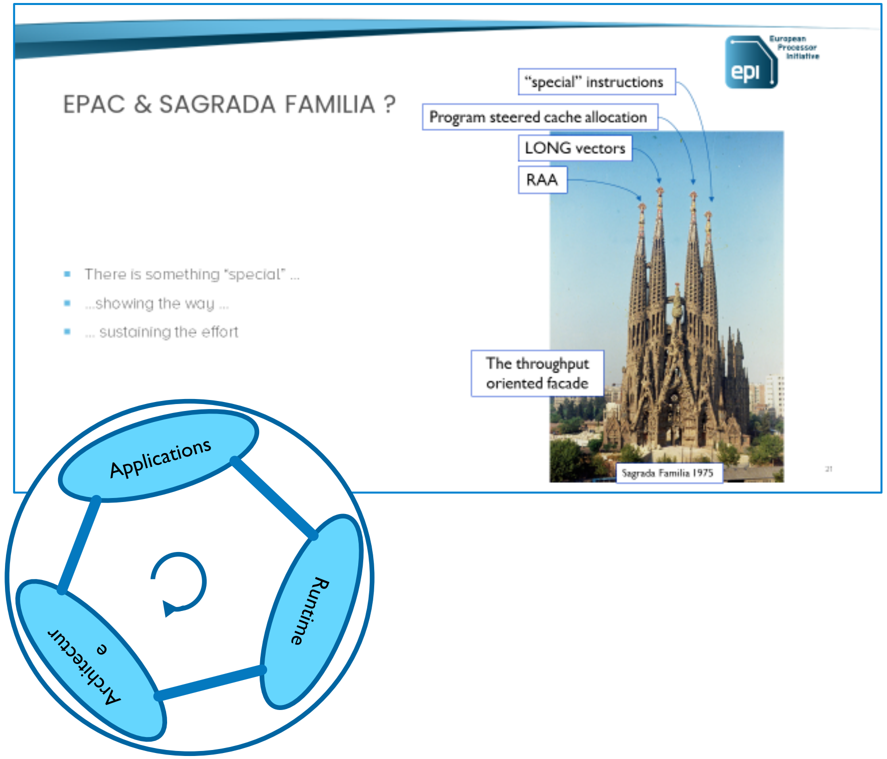
- SDV3
  - High end FPGA implementation
  - Standalone EPAC
  - HBM
  - Software development.
  - Flexible architectural early extension
- Test board for EPAC Test chip
- Integration in automotive MCP demonstrator



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



EPAC & SAGRADA FAMILIA ?

- There is something "special" ...
- ...showing the way ...
- ... sustaining the effort

"special" instructions

Program steered cache allocation

LONG vectors

RAA

The throughput oriented facade

Sagrada Familia 1975

21