Reconfigurable Computing in the EuroEXA Project

Presenter – Paul Carpenter

Barcelona Supercomputing Center

paul.carpenter@bsc.es

Leader WP2 Applications and Software Technical Manager, EuroEXA

Leader WP3 Enablement of Software Compute Node, ExaNoDe

This research is supported by the European Commission under the "Horizon 2020 Framework Programme" with grant number 754337

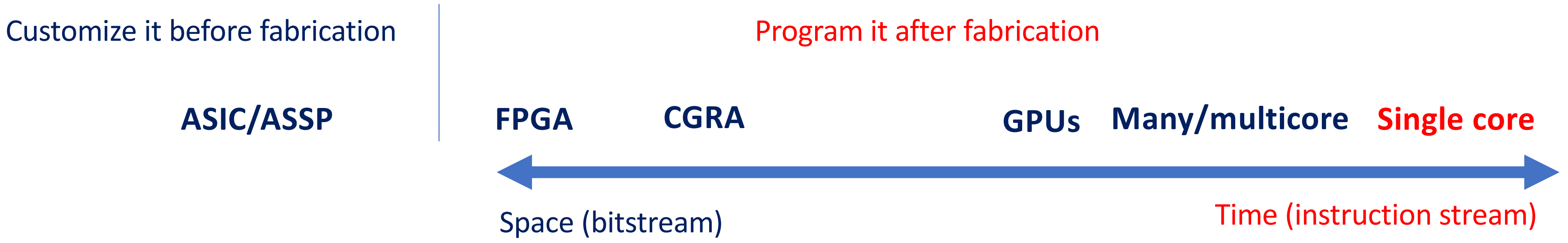
Disclaimer

- Many people here may know more about reconfigurable computing than I do
- I am here to learn as much as to present…

- Acknowledgments to the EuroEXA, AXIOM, LEGaTO, EPEEC project partners
- Special thanks to the BSC FPGA team: Xavier Martorell, Daniel Jiménez, Carlos Alvarez, Antonio Filgueras, Miquel Vidal, Jaume Bosch... and the BSC PM team

Tipping point for reconfigurable computing?

- *“Transistors/vacuum tubes are expensive so I can only build a single core”*
  - Share arithmetic unit by multiplexing with instructions in time
  - => **Single core** (von Neumann architecture)



Customize it before fabrication

Program it after fabrication

ASIC/ASSP

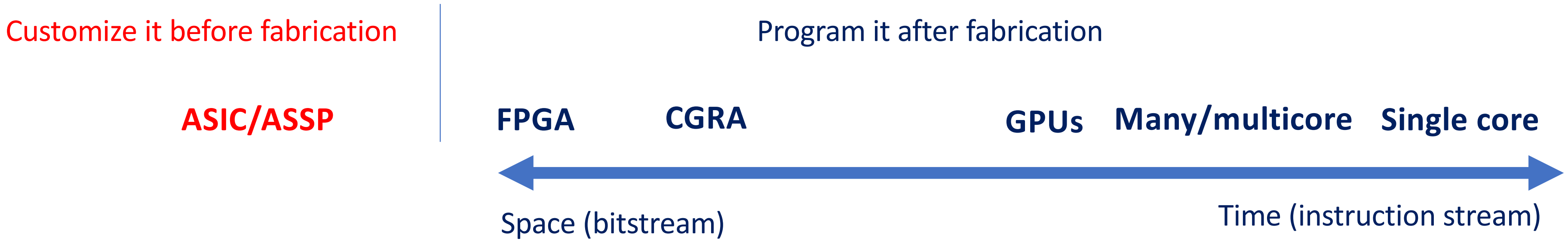
FPGA **CGRA** **GPUs** **Many/multicore** **Single core**

Space (bitstream)

Time (instruction stream)

Tipping point for reconfigurable computing?

- *“I know what I want it to do”*
  - Design it before fabrication and for a specific purpose
  - Unless volumes are extremely high you will need use an older technology node
  - **=> ASIC/ASSP**



Customize it before fabrication

ASIC/ASSP

Program it after fabrication

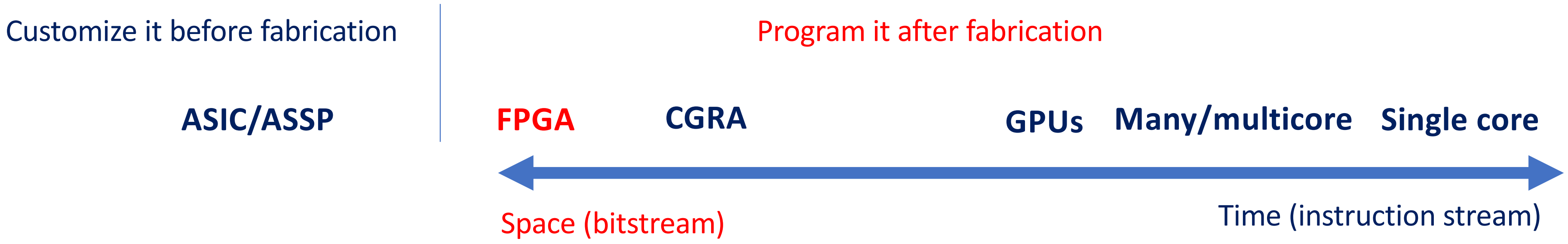
FPGA **CGRA** **GPUs** **Many/multicore** **Single core**

Space (bitstream)

Time (instruction stream)

Tipping point for reconfigurable computing?

- End of Dennard scaling: *“Transistors are cheap and the problem is energy"*
  - Assumption behind von Neumann is no longer applicable
  - Why build a complex control unit to multiplex a small number of execution units?
  - No control unit => power efficiency much better than a CPU
  - **=> FPGA**: reconfigurable in space


Customize it before fabrication
ASIC/ASSP
Program it after fabrication
FPGA
CGRA
GPUs
Many/multicore
Single core
Space (bitstream)
Time (instruction stream)

EUROEXA
FUNDED BY THE EUROPEAN UNION

EuroEXA project

EURO EXA
FUNDED BY THE EUROPEAN UNION

Commercial Partners

ICEOTOPE
SYNELIXIS
arm
MAXELER Technologies
Maximum Performance Computing
ZeroPoint Technologies

Academic/Gov. Partners

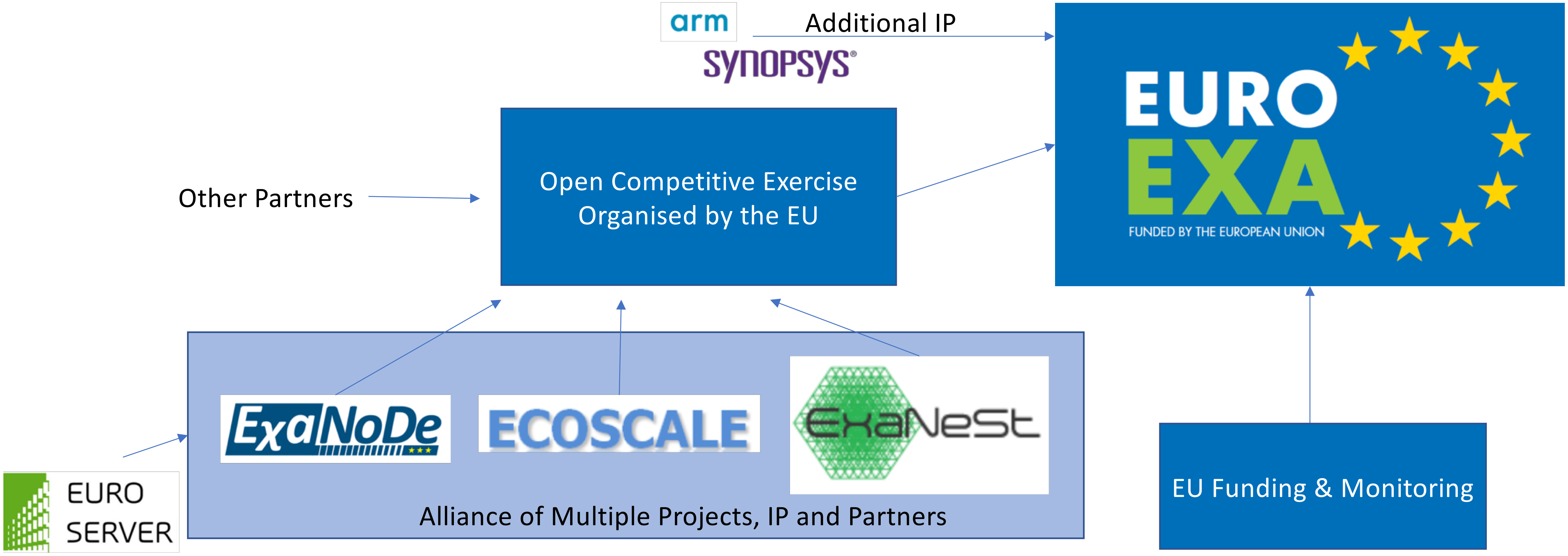
ICCS
MANCHESTER 1824
The University of Manchester
INAF ISTITUTO NAZIONALE DI ASTROFISICA
BSC Barcelona Supercomputing Center
Centro Nacional de Supercomputación
imec
FORTH
FOUNDATION FOR RESEARCH AND TECHNOLOGY - HELLAS
Fraunhofer
Hartree Centre
Science & Technology Facilities Council
ECMWF
neurasmus
The Erasmus Neuroscience Company
INFN
Istituto Nazionale di Fisica Nucleare

Supporters

XILINX
SYNOPSYS®
Silicon to Software™
SKA
SQUARE KILOMETRE ARRAY
Met Office
cnag
centre nacional d'anàlisi genòmica
centro nacional de análisis genómico
janssen
NYU School of Medicine
NYU LANGONE MEDICAL CENTER
CRS4
IDEAS BECOME LIFE
Schneider Electric
Noldus
Information Technology
SOLVAY
asking more from chemistry®

- H2020 FETHPC-01-2016
- September 2017 – February 2021 (3.5 years)
- Budget €20 M

EuroEXA in context



Additional IP

Open Competitive Exercise Organised by the EU

Other Partners

Alliance of Multiple Projects, IP and Partners

EU Funding & Monitoring

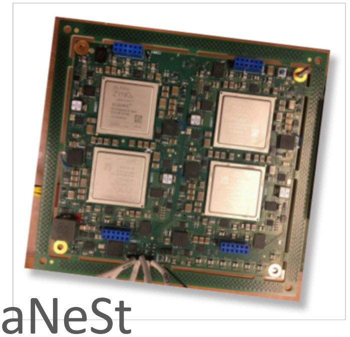
Our Vision

- Testbed architecture will be shown to be capable of scaling to world-class **peak performance in excess of 400 PFLOPS** with an estimated **system power of around 30 MW peak.**
- **A compute-centric 250 PFLOPS per 15 MW** by 2019.
- **Show that an exascale** machine could be built in 2020 within 30 shipping containers with a edge to edge distance of less than 40 m

Co-design and demonstrate 1 PF+ testbed in operational environment

Three testbeds to be deployed at STFC, Daresbury

Testbed 1

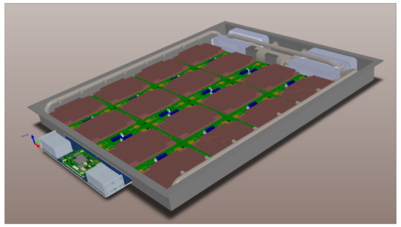


50 nodes of ExaNeSt technology for **software development**

Testbed 2

~500 co-designed nodes and new infrastructure technologies to **test scaling**

Testbed 3



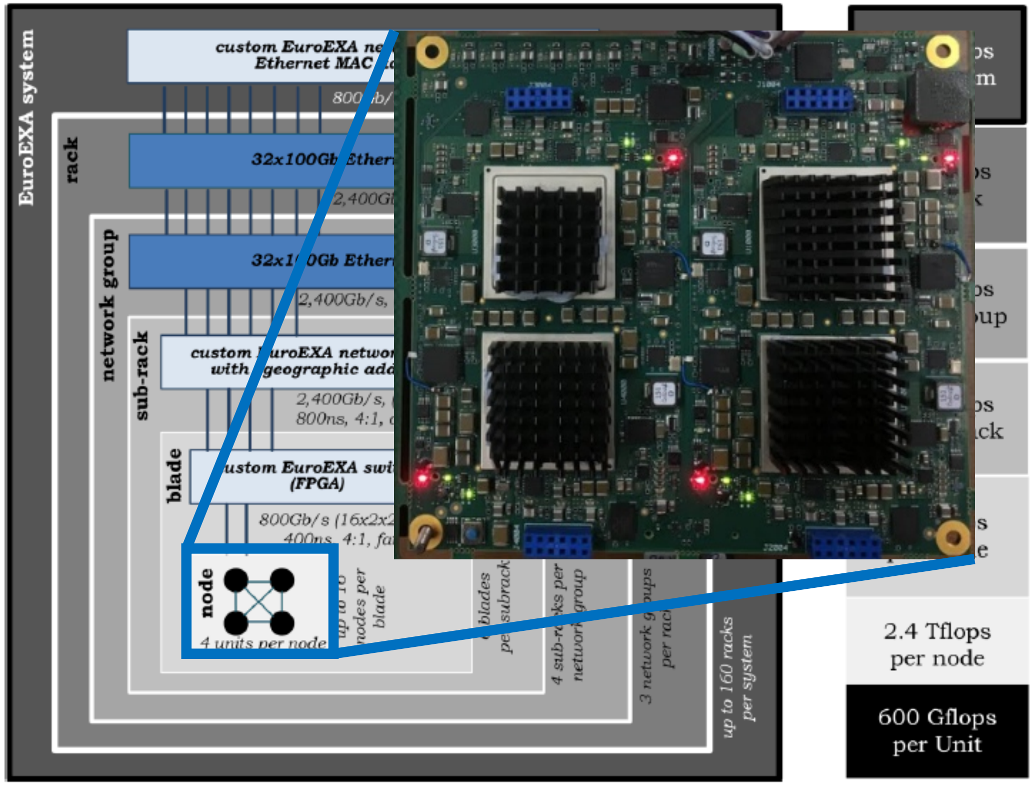
Test **new node and processor technologies** that will ultimately project to exascale

Mid 2018

Early 2019

2020

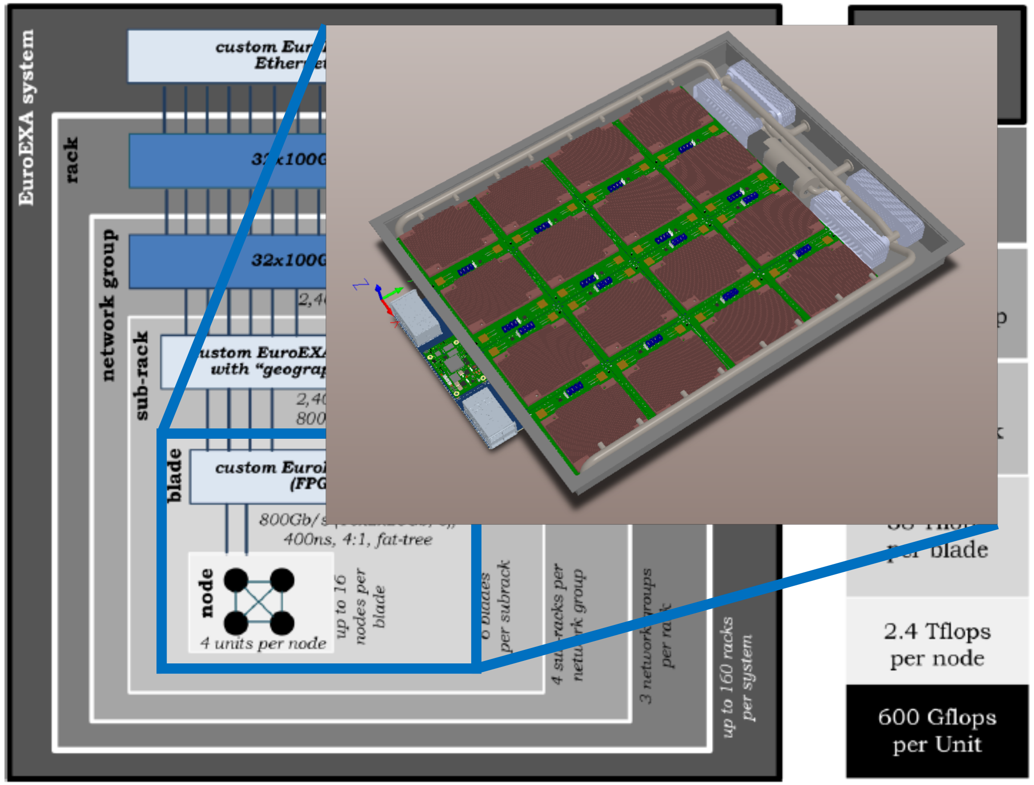
System architecture and technology: Testbed 1 compute node



Initial technology from FORTH (QFDB)

- 12 cm x 13 cm
- 4x Zynq UltraScale+
  - 4 ARM Processors and 4 FPGA Accelerators
- M.2 SSD
- 4 x SODIMMs + Onboard RAM
- Daughterboard style
- 160 Gb/s of I/O

System architecture and technology: Blades



Technology from Iceotope:

- 16 Node half depth 1u chassis
- 2 x 3.2 kW per U (back2back)
- Total Liquid Cooling technology
- 48 V DC distribution
- Hot water out, chiller-less operation

Co-design for Testbed 2 and 3

=> ZU9 for acceleration FPGA is too small!

Original TB2/3 plan: QFDB

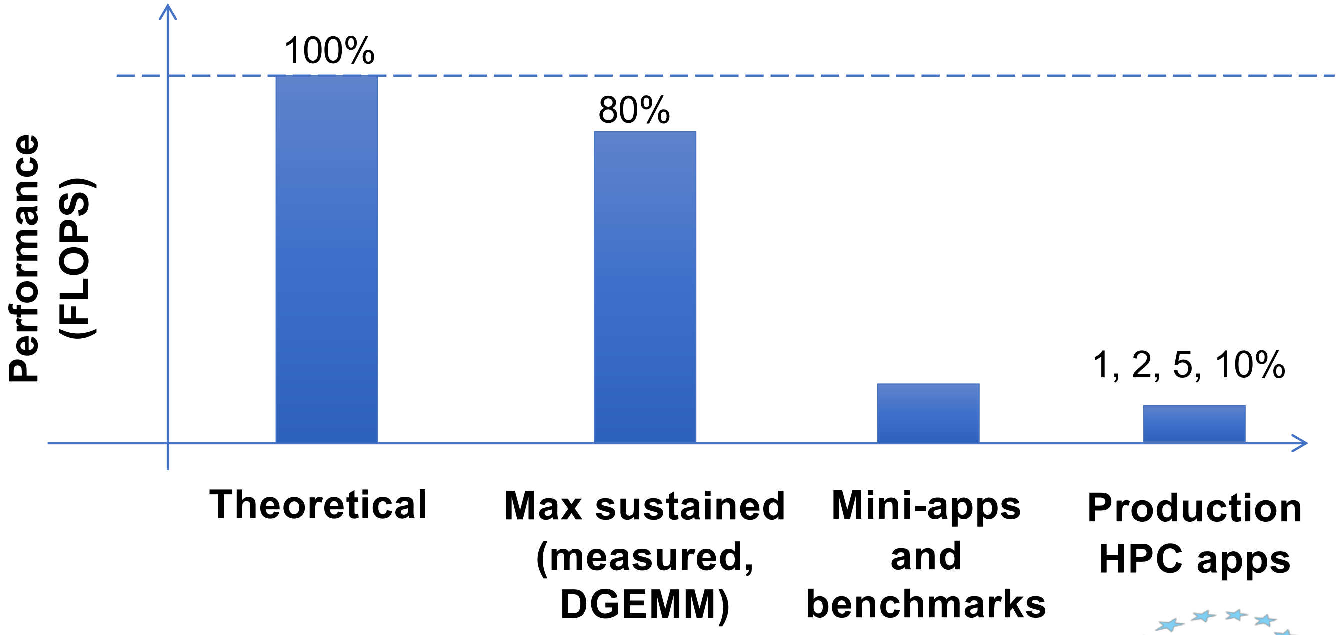
Four Xilinx Zynq UltraScale+ ZU9P

New TB2/3 proposal: CRDB

Xilinx Zynq UltraScale+ ZU9P for interconnect

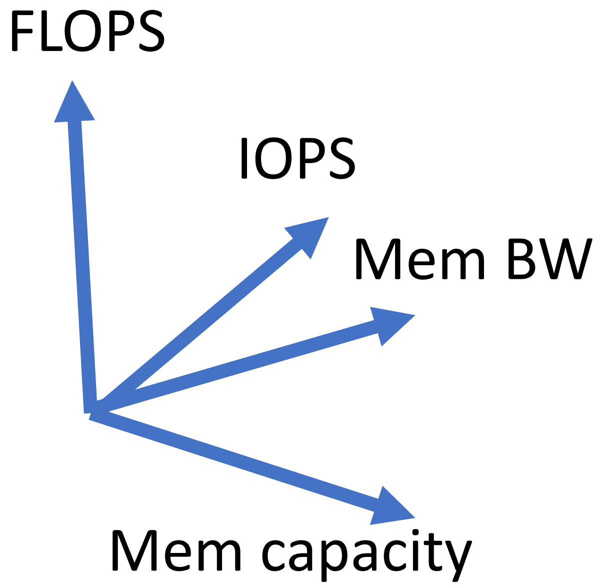
Xilinx Zynq UltraScale+ VU9P for acceleration

Why focus on applications? Theory vs. Practice



- e.g. Sunway's TaihuLight
- 93 PF and 6 GF/W for HPL
- But 0.3% of peak performance for more realistic HPCG

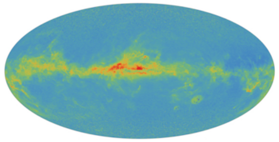
Deliver available performance to full-scale production applications

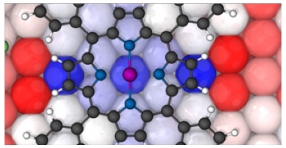

FLOPS

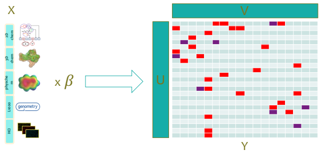
IOPS

Mem BW

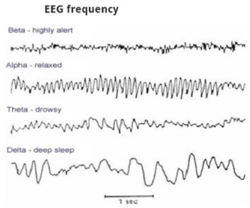
Mem capacity


AVU-GSR

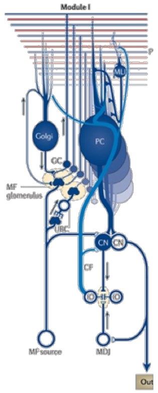

Quantum Espresso



SMURFF

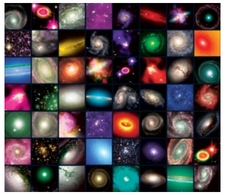


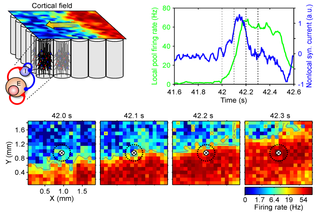
Neuromarketing



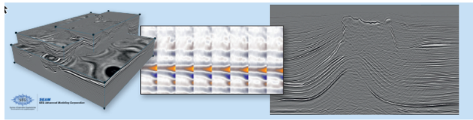
InfOli

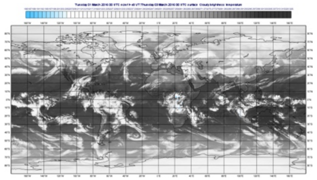
NEMO

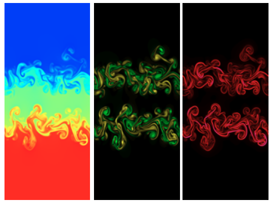

Astronomy image classification

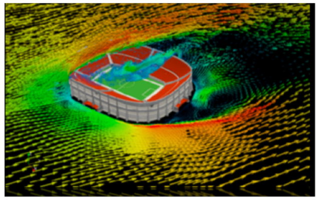


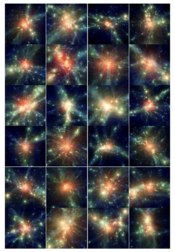
NEST/DPSNN

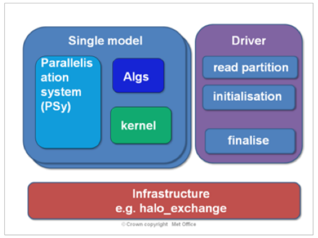

FRTM


IFS


LBM


Alya


GADGET


Single model

Parallelisation system (PSy)

Algs

kernel

Driver

read partition

initialisation

finalise

Infrastructure e.g. halo_exchange

LFRic

Why these 14 applications?

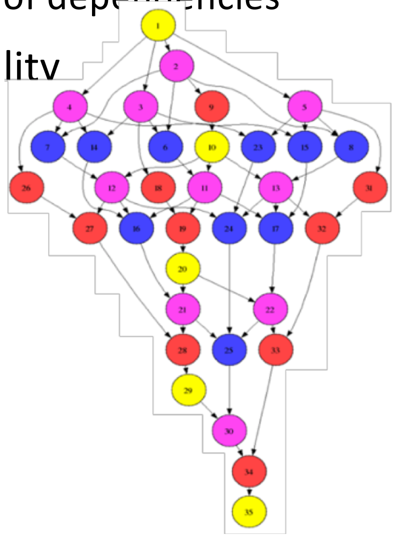
- 10 strong European application partners
- 14 applications covering three important application domains
  - Climate and weather (LFRic, NEMO, IFS)
  - Physics and energy (LBM, Alya, GADGET, AVU-GSR, FRTM, Astronomy image classification, Quantum E)
  - Life science and bioinformatics (NEST, Neuromarketing, InfOli, SMURFF)
- These domains will require exascale computing in near future
- ALYA, GADGET, NEMO and Quantum ESPRESSO part of PRACE UEABS
- Applications for porting, demonstration, evaluation and co-design

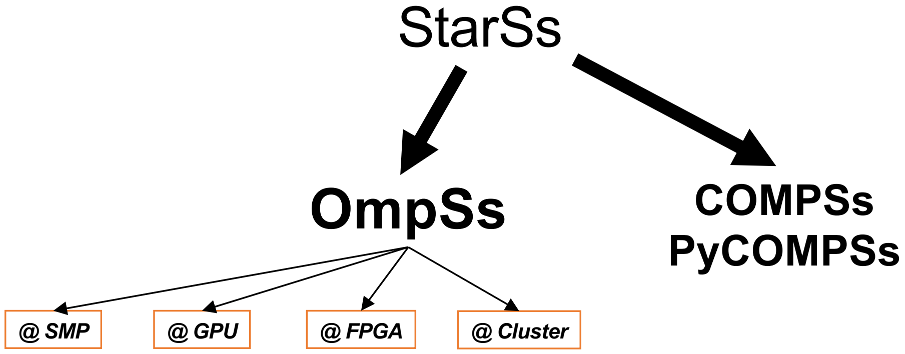
EuroEXA programming environment

- EuroEXA has unique hardware capabilities
  - FPGA acceleration
  - UNIMEM communications
- Applications will target portable programming models / communication libraries
  - MPI, GPI, OpenStream, OmpSs, MaxJava and PSyClone
- Optimization for EuroEXA architecture done by programming environment
  - Building on the work in previous projects: ExaNoDe, AXIOM, INTERTWINE, …
  - EuroEXA brings greater maturity, especially in FPGA support
  - Tuning and optimization through EuroEXA's full applications
- **New from point of view of implementation: integration is very challenging!**

Deep dive into one programming model: OmpSs

- StarSs family: key concepts
  - Sequential task-based program
  - View of a single address/name space
  - Happens to execute in parallel: automatic runtime computation of dependencies
- Productivity and portability





- **Long-term research at BSC**
  - 10+ years' investment
  - Tasking, task dependencies, heterogeneity, multiple address spaces, etc.
- **Pushing ideas to OpenMP standard**
  - Task dependencies in OpenMP 4.0
  - Priorities in OpenMP 4.5
  - Task reductions and multidependences in OpenMP 5.0
- **Mercurium compiler**
- **Nanos runtime system**

Goals of OmpSs@FPGA

- Add toolchain support for parallel programming on FPGAs
- Allowing the use of (heterogeneous) task parallelism both in the CPUs and the FPGAs
  - Including support for task dependences (OpenMP 4.0-4.5-5.0)
- Automating code generation of the CPU and FPGA binaries
  - Transparently running open or vendor tools
- Automating memory allocation and data copies
- Provide facilities to perform blocking from inside the FPGA accelerators
- Provide FPGA execution tracing facilities and visualization support

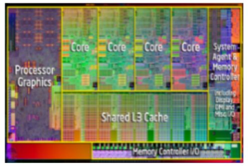
Heterogeneous programming

- The “implements” approach

```
#pragma omp target device(fpga) implements(matrix_multiply)
#pragma omp task in([BS]a,[BS]b) inout([BS]c)
void  matrix_multiply_fpga(float a[BS][BS], float b[BS][BS],
                                            float c[BS][BS]);
```

FPGA

```
#pragma omp target device(smp) copy_deps
#pragma omp task in([BS]a,[BS]b) inout([BS]c)
void matrix_multiply(float a[BS][BS], float b[BS][BS],
                                      float c[BS][BS]);
```



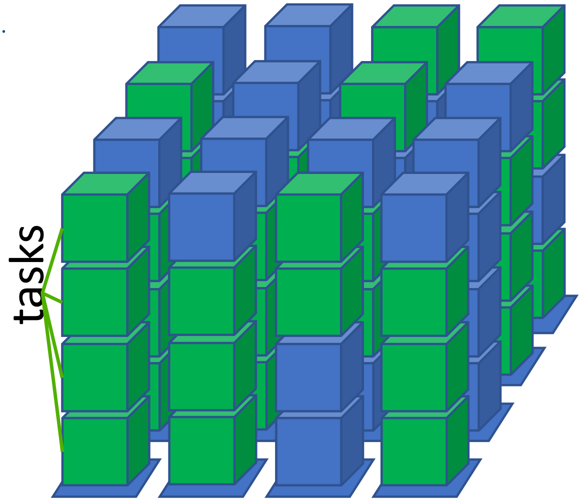
SMP

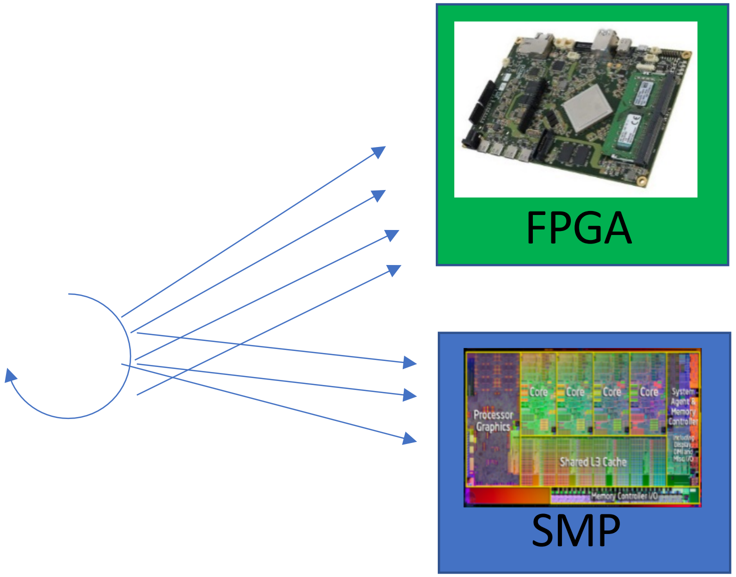
```
…
for (i=0; i<NB; i++)
  for (j=0; j<NB; j++)
    for (k=0; k<NB; k++)
       matrix_multiply(A[i][k], B[k][j], C[i][j]);
…
```

Heterogeneous programming

- The “implements” approach

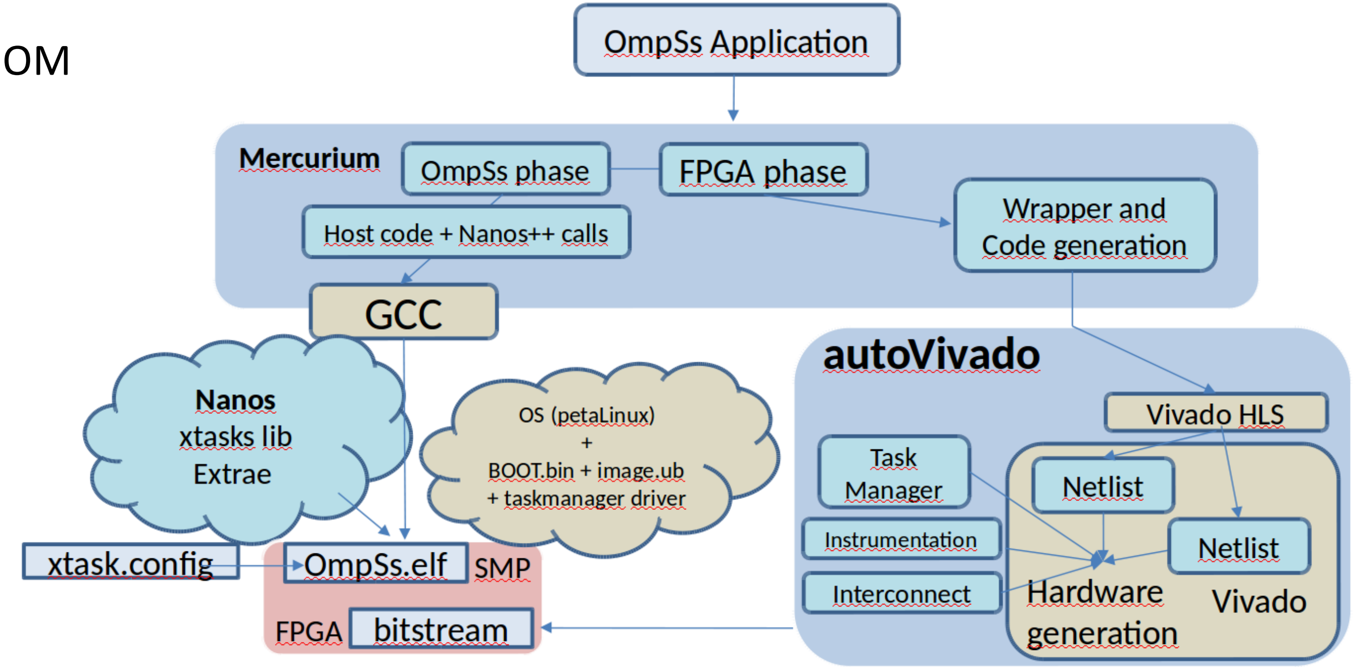
```
for (i=0; i<NB; i++)
  for (j=0; j<NB; j++)
    for (k=0; k<NB; k++)
       matrix_multiply(A[i][k], B[k][j], C[i][j]);
…
```





Automated compilation

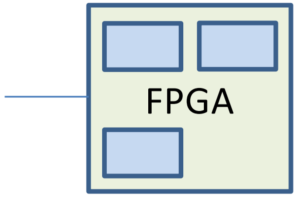
- Currently targets Xilinx FPGAs
- Zedboard, 702, 706, Zynq U+ EG9 (AXIOM Board), ZCU102
- Currently adding new products
  - Zybo, COM Express, Zynq U+ ZU3, Alpha-Data
- TODO:
  - QFDB ongoing
  - FPGA Cloud
  - Xilinx Alveo



Automated compilation

- Transforms directives onto source code for autoVivado

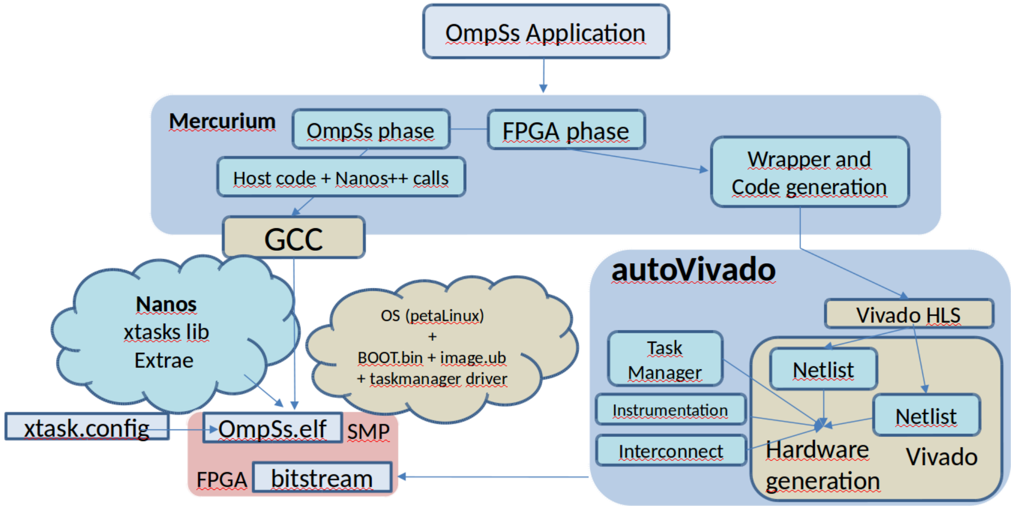
```
#pragma omp target device(fpga) copy_deps num_instances(3)
#pragma omp task in([BSIZE]a, [BSIZE]b) inout([BSIZE]c)
void matmulBlock(elem_t a[BSIZE][BSIZE], elem_t b[BSIZE][BSIZE], elem_t
c[BSIZE][BSIZE]);
```



`elem_t` represents a value

In the examples:

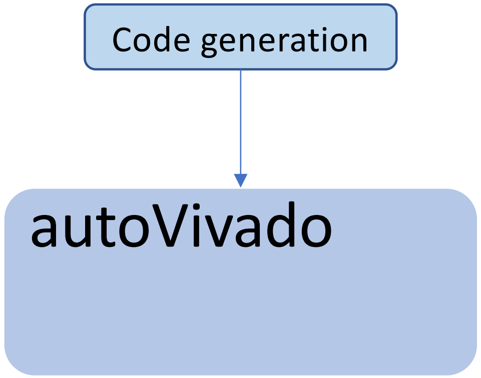
```
    typedef float elem_t;
```



Automating memory allocation and data transfers

- We get a wrapper function with the user's source code inlined
  - Inline directive

```
void matmulBlock(elem_t a[BSIZE][BSIZE], elem_t b[BSIZE][BSIZE], elem_t c[BSIZE][BSIZE])
{
#pragma HLS inline
       // as written by the programmer
}

// automatically generated
void matmulBlock_hls_automatic_mcxx_wrapper(
       hls::stream<axiData> &inStream,            // standard in and out streams
       hls::stream<axiData> &outStream,
       uint8_t accID,                                       // to signal termination
       elem_t (*mcxx_a), elem_t (*mcxx_b), elem_t (*mcxx_c)) {

       // mcxx_a, mcxx_b and mcxx_c are ports from where to copy
       // external data
```

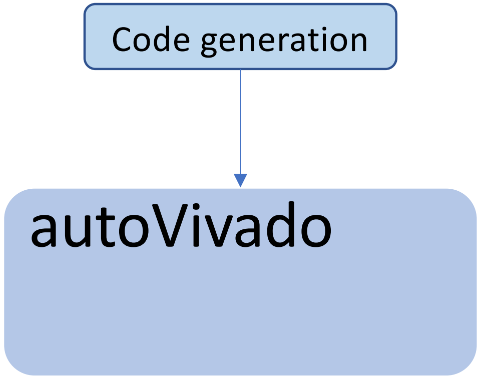

Code generation

autoVivado

Automating memory allocation and data transfers

- Saves the programmer from error-prone coding

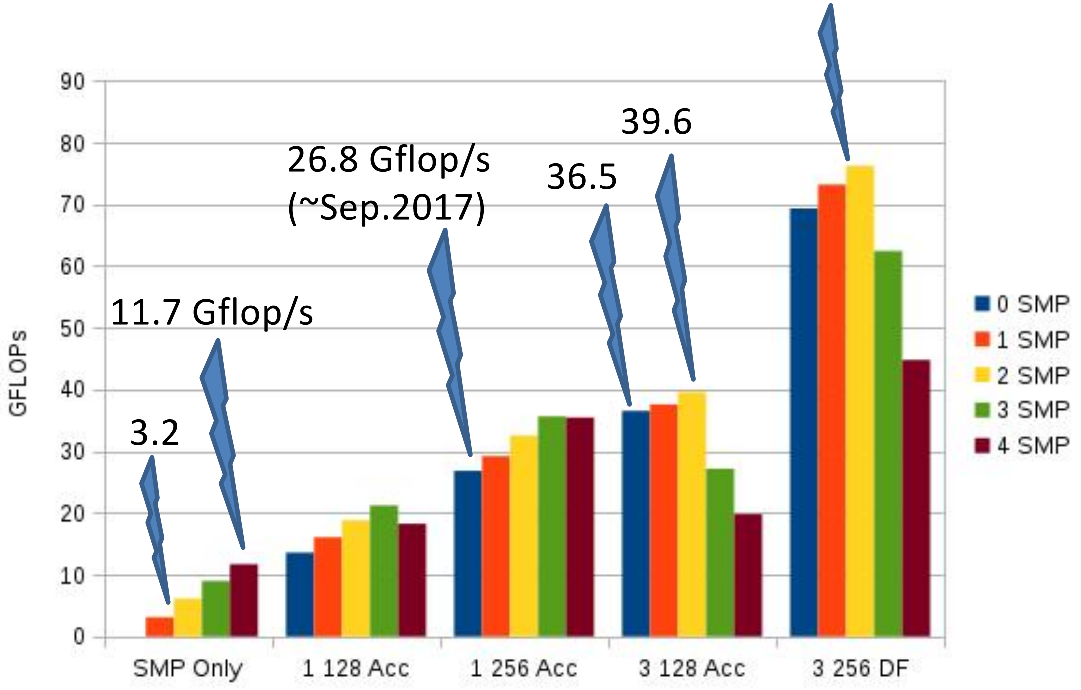
```
…
    // mcxx_a, mcxx_b and mcxx_c are ports from where to copy external data

        // declare data in BLOCK-RAM
        static elem_t a[BSIZE][BSIZE], b[BSIZE][BSIZE], c[BSIZE][BSIZE];
        ...

        // copy from external memory to BLOCK_RAM
        memcpy (a, mcxx_a + inStream.read().data, BSIZE*BSIZE);
        ...

        // Execute the accelerator
        matmulBlock(a, b, c);

        // same procedure to memcpy to external memory
        memcpy (mcxx_c + ...
}
```



How this translates to the Zynq UltraScale+

AXIOM SECO board

- Matrix multiply 2048x2048 single precision
- Implements
  - SMP OpenBLAS
  - Great success
- 300 MHz on FPGA
  - Outperforms cores
- Parallel creation
- New dataflow kernel



(*)76.0 Gflop/s

39.6

26.8 Gflop/s (~Sep.2017)

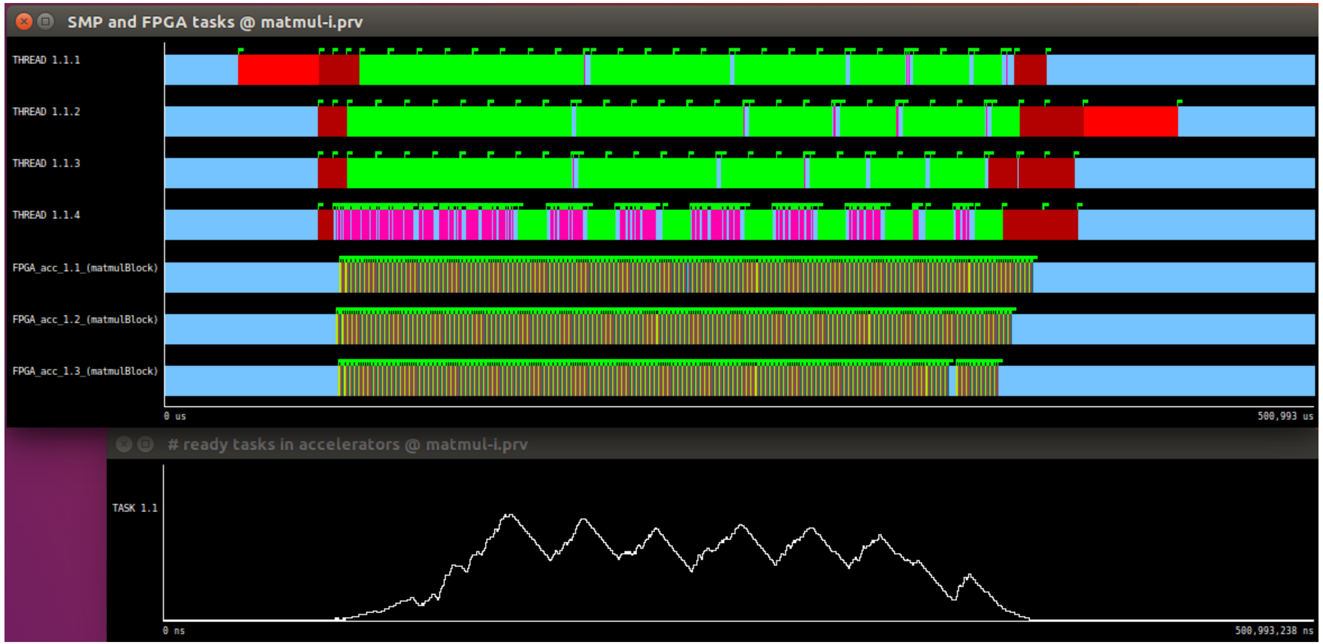
36.5

11.7 Gflop/s

3.2

Instrumentation on Zynq U+

- 3x256 DF with implements (4 workers)
  - Cores do a lot of work (part of the computation)


SMP and FPGA tasks @ matmul-i.prv
THREAD 1.1.1
THREAD 1.1.2
THREAD 1.1.3
THREAD 1.1.4
FPGA_acc_1.1_(matmulBlock)
FPGA_acc_1.2_(matmulBlock)
FPGA_acc_1.3_(matmulBlock)
0 us
500,993 us
# ready tasks in accelerators @ matmul-i.prv
TASK 1.1
0 ns
500,993,238 ns

OmpSs@FPGA conclusions

- Programming models should…
  - Enable productivity and portability
  - Support for heterogeneous/hierarchical architectures
  - Support asynchrony → global synchronization is not an answer anymore
  - Be aware of data locality
- OmpSs proposal enables…
  - Incremental parallelization from existing sequential codes
  - Data-flow execution model that naturally supports asynchrony
  - Nicely integrates heterogeneity and hierarchy
    - On GPUs, FPGAs, towards a Cloud of FPGAs
  - Support for locality scheduling, diverse implementations
  - Active and open source project, please find more information/downloads

http://pm.bsc.es

Xavier Martorell

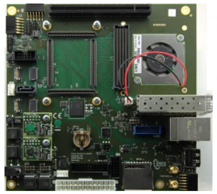
Ongoing work in OmpSs@FPGA

- Keep giving support to more integrated and discrete FPGAs

```
#define BS 128

void matrix_multiply(float a[BS][BS], float b[BS][BS],float c[BS][BS])
{
#pragma HLS inline
 int const FACTOR = BS/2;
#pragma HLS array_partition variable=a block factor=FACTOR dim=2
#pragma HLS array_partition variable=b block factor=FACTOR dim=1
 // matrix multiplication of a A*B matrix
 for (int ia = 0; ia < BS; ++ia)
  for (int ib = 0; ib < BS; ++ib) {
#pragma HLS PIPELINE II=1
   float sum = 0;
   for (int id = 0; id < BS; ++id)
    sum += a[ia][id] * b[id][ib];
   c[ia][ib] += sum;
  }
}
```

Xilinx Alveo

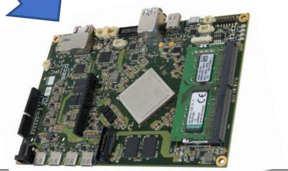
Alpha-Data



EuroEXA prototypes

Trenz Electronics Zynq U+
TE0808 XCZU9EG-ES1

4x Cortex-A53 cores + FPGA
(64-bit platforms)



SECO AXIOM Board
Zynq U+ XCZU9EG-ES2

Zynq-7000 Family
2x Cortex-A9 cores + FPGA
(32-bit platforms)

Recap…

- End of Dennard scaling => transistors are cheap and the problem is energy
- von Neumann optimizes for the wrong thing (expensive transistors)

- => Growing pressure towards reconfigurable computing and FPGAs

- EuroEXA is ahead of the curve
  - Building large-scale FPGA-accelerated prototype
  - Porting 14 applications to use FPGA acceleration via runtime systems

Challenges

- Still research topic: multiple approaches being pursued
  - In EuroEXA: Accelerated libraries, OpenStream, OmpSs, ECOSCALE, Maxeler MaxJava, PSyClone

- Personal view: transition to reconfigurable computing will take a decade or more
  - Challenges in programming models, compilers, architectures
  - Current programming approaches have focussed on von Neumann for ~50 years
    - Layers of abstractions for sequential programming: languages, compilers, DLLs, operating systems, APIs, etc.
  - Lack of higher level programming abstractions for reconfigurable computing
  - Excessive compilation time (place and route) stifles programming productivity
  - Missing functionality for HPC: double precision floating point

Summary

- EuroEXA will co-design and demonstrate 1 PF+ testbed in operational environment
- Vision is exascale machine for Europe

- High performance for full-scale production applications
- Co-design led to SoC design choices and change in FPGA
- Complete software stack and optimized portable programming models
- Ongoing porting of applications to the architecture