LEGaTO: First Steps Towards Energy-Efficient Toolset for Heterogeneous Computing

SAMOS XVIII

Tobias Becker (Maxeler)

18/July/2018

The LEGaTO project has received funding from the European Union's Horizon 2020 research and innovation programme under the grant agreement No 780681

The industry challenge

- Dennard scaling is dead
- Moore's law is slowing down

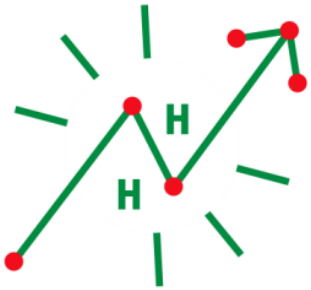
- Trend towards heterogeneous architectures
- Increasing focus on energy: ICT sector is responsible for 5% of global energy consumption

- Creates a programmability challenge

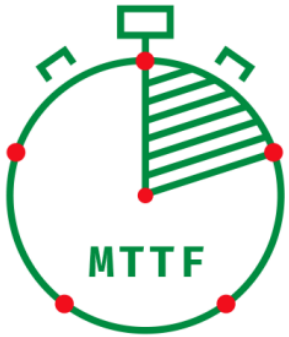
LEGaTO Ambition

- Create software stack-support for energy-efficient heterogeneous computing
  - Starting with Made-in-Europe mature software stack, and optimizing this stack to support energy-efficiency
  - Computing on a commercial cutting-edge European-developed heterogeneous hardware substrate with CPU + GPU + FPGA + FPGA-based Dataflow Engines (DFE)
- Main goal: energy efficiency

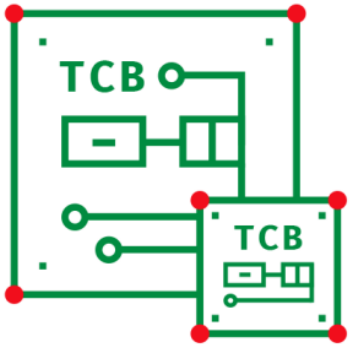
LEGaTO Objectives



One order of magnitude improvement in energy-efficiency for heterogeneous hardware through the use of the energy-optimized programming model and runtime.



5× decrease in Mean Time to Failure through energy-efficient software-based fault tolerance.



Size reduction of the trusted computing base by at least an order of magnitude.

5× increase in FPGA designer productivity through the design of novel features for hardware design using dataflow languages.

Partners

- BSC (Barcelona Supercomputing Center)
- CHALMERS (Chalmers University of Technology)
- UNINE (Neuchatel University)
- TUD (Technical University of Dresden)
- CHR (Christmann)
- UNIBI (University of Bielefeld)
- TECHNION (Technion, Israel Institute of Technology)
- MAXELER (Maxeler Technologies Limited)
- DIS (Data Intelligence Sweden)
- HZI (Helmholtz Centre for Infection Research)

- Duration: Dec 2017 – Nov 2020

Overview

Smart Home/City

Healthcare

Machine Learning

OmpSs

Fault-tolerance

Task-based PM and runtime

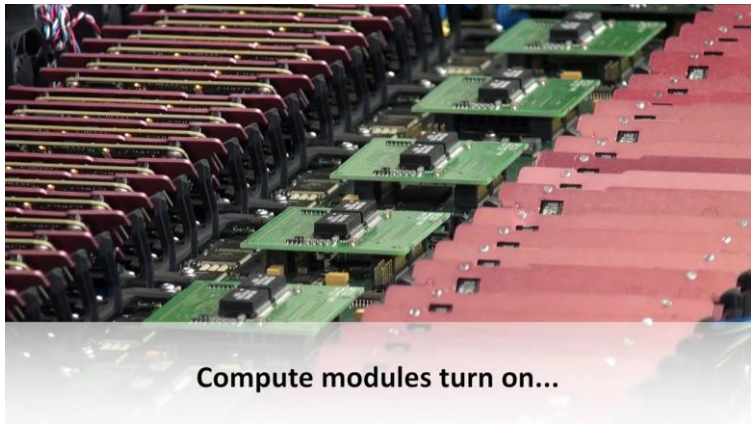
Productivity

Security

Heterogeneous Hardware Platforms

Hardware Platforms

- Christmann (RECS Box)
  - Heterogeneous platform: Supports CPU (X86 or Arm), GPU (Nvidia) and FPGA (Xillinx and Altera)

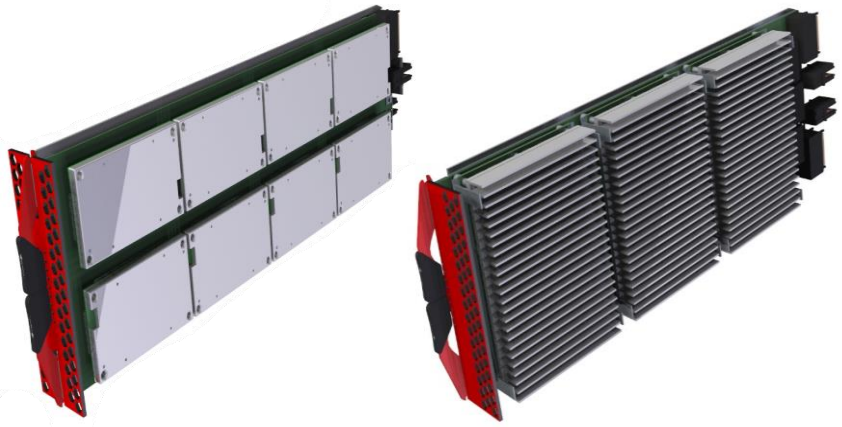


- MAXELER
  - Maxeler DFE platforms and toolset

RECS|Box overview

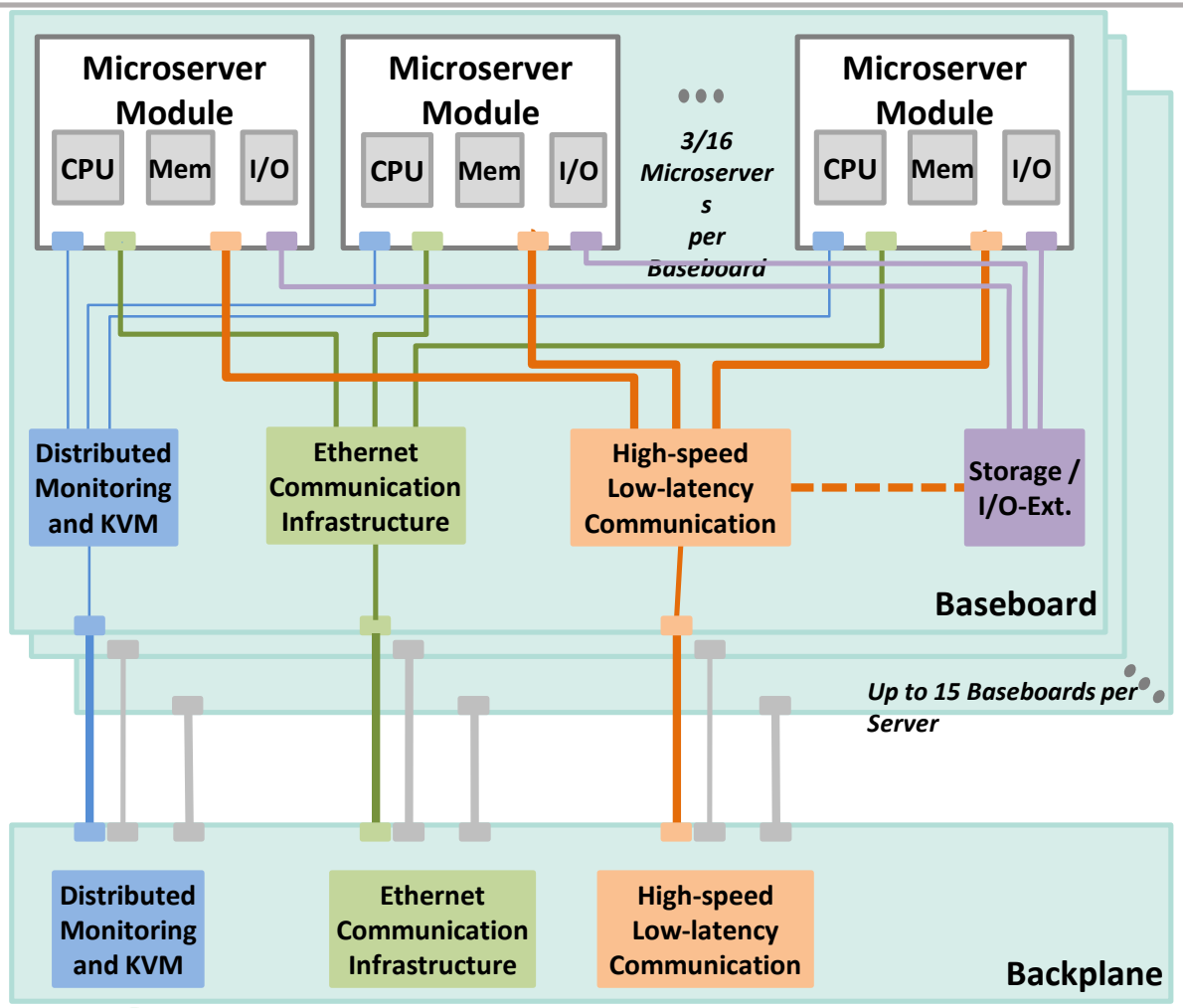
Concept

- Heterogeneous microserver approach, integrating CPU, GPU and FPGA technology
- High-speed, low latency communication infrastructure, enabling accelerators, scalable across multiple servers
- Modular, blade-style design, hot pluggable/swappable



RECS|Box overview

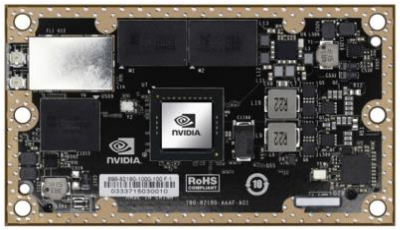
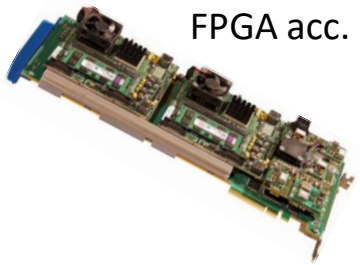
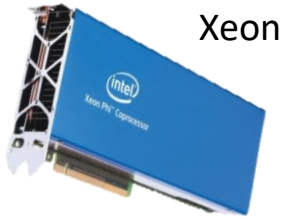
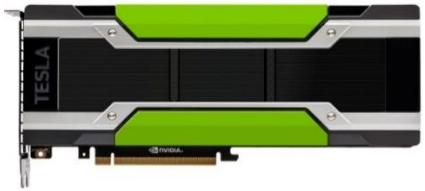
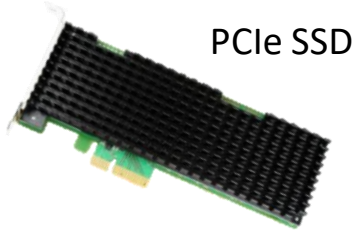
Architecture



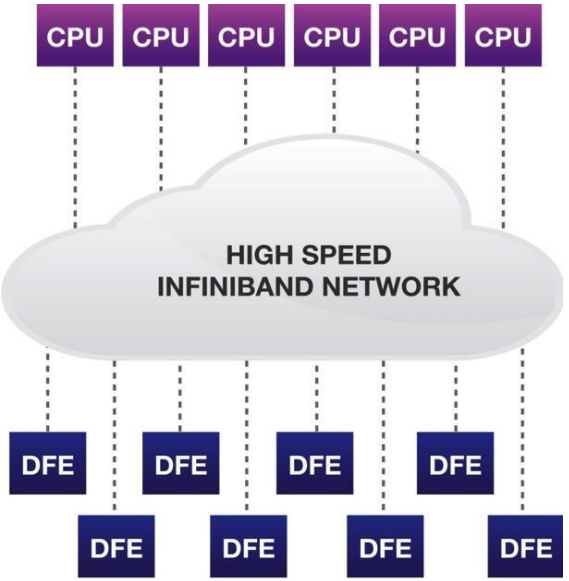
- Dedicated network for monitoring, control and iKVM
- Multiple 1Gb/10Gb Ethernet links per Microserver – integrated 40 Gb Ethernet backbone
- Dedicated high-speed, low-latency communication infrastructure
  - Host-2-Host PCIe between CPUs
  - Switched high-speed serial lanes between FPGAs
  - Enables pooling of accelerators and assignment to different hosts

RECS|Box overview

Microserver

	CPU Microserver	GPGPU	FPGA Microserver	PCIe-Extensions
Low-Power Microserver				RAPTOR FPGA acc. 
	NVIDIA Jetson TX1/TX2 4 Core A57@1.73 GHz + Maxwell GPGPU@1.5 GHz 2 Core Denver + 4 Core A57 + Pascal GPGPU@1.5 GHz		Xilinx Zynq 7020 85 kLC	Xeon Phi 
High-Performance Microserver				PCIe SSD 
	ARMv8 Server SoC 32 Core A72@2.1 GHz	NVIDIA Tesla P100/V100 Pascal/Volta GPGPU@1.3 GHz	Intel Stratix 10 SoC 2,800 kLE	

Maxeler Dataflow Computing Systems

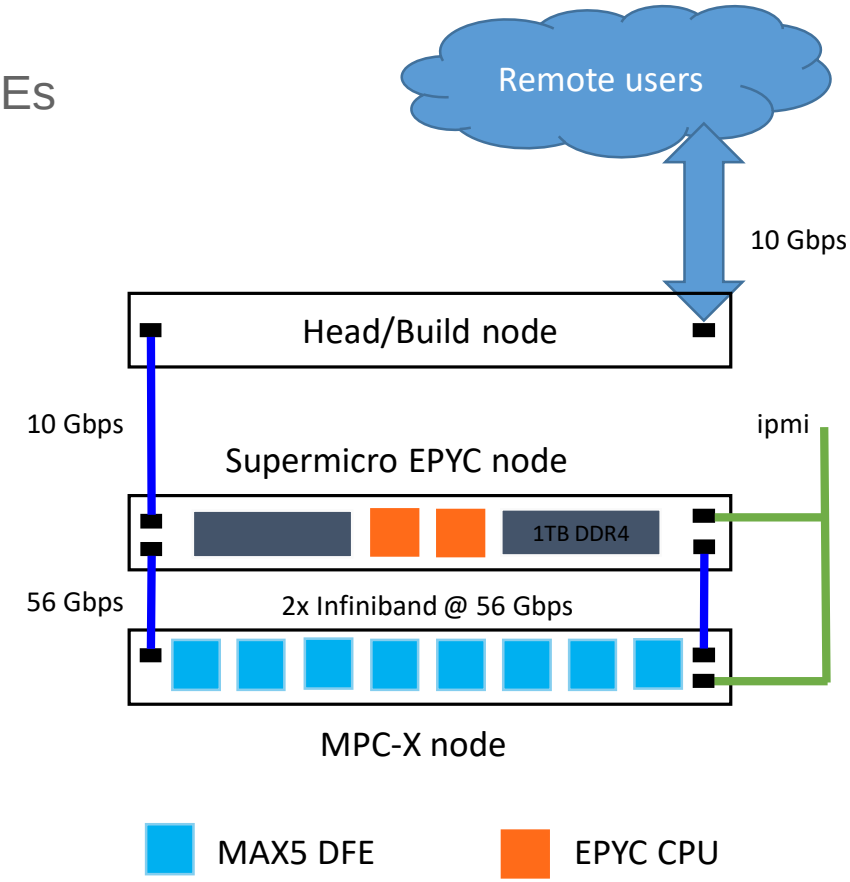


MPC-X3000

- 8 Dataflow Engines in 1U
- Up to 1TB of DFE RAM
- Zero-copy RDMA
- Dynamic allocation of DFEs to conventional CPU servers through Infiniband
- Equivalent performance to 20-50 x86 servers

Dataflow System at Jülich Supercomputing Center

- Pilot system deployed in October 2017 as part of PRACE PCP
- System configuration
  - 1U Maxeler MPC-X with 8 MAX5 DFEs
  - 1U AMD EPYC server
    - 2x AMD 7601 EPYC CPUs
    - 1 TB RAM
    - 9.6 TB SSD (data storage)
  - one 1U login head node
- Runs 4 scientific workloads



European Commission

PRACE

http://www.prace-ri.eu/pcp/

Scaling Dataflow Computing into the Cloud

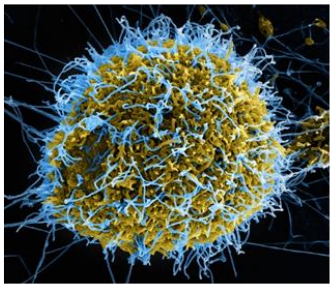
On-premise

- MAX5 DFEs are compatible with Amazon EC2 F1 instances
- MaxCompiler supports programming F1
- Hybrid Cloud Model:
  - On-premise Dataflow system from Maxeler
  - Elastically expand to the Amazon Cloud when needed

Cloud

Use Cases

- Healthcare: Infection biomarkers
  - Statistical search for biomarkers, which often needs intensive computation. A biomarker is a measurable value that can indicate the state of an organism, and is often the presence, absence or severity of a specific disease



Infection research **(HZI)**

- Smart Home: Assisted Living
  - The ability of the home to learn from the users behavior and anticipate future behavior is still an open task and necessary to obtain a broad user acceptance of assisted living in the general public

Smart home (**UNIBI**)

Use Cases

- Smart City: operational urban pollutant dispersion modelling
  - Modeling city landscape + sensor data + wind prediction to issue a “pollutant weather prediction”

Smart city **(BSC)**

- Machine Learning: Automated driving and graphics rendering
  - Object detection using CNN networks for automated driving systems and CNN- and LSTM-based methods for realistic rendering of graphics for gaming and multi-camera systems

Machine learning **(DIS)**

- Secure IoT Gateway
  - Variety of sensors and actors in an industrial and private surrounding

Secure IoT gateway **(CHR)**

Programming Models and Runtimes: Task based programming + Dataflow!

- OmpSs / OmpSs@FPGA (BSC)
- MaxCompiler (Maxeler)
- DFiant HLS (TECHNION)

- XiTAO (CHALMERS)

Towards a single source for any target

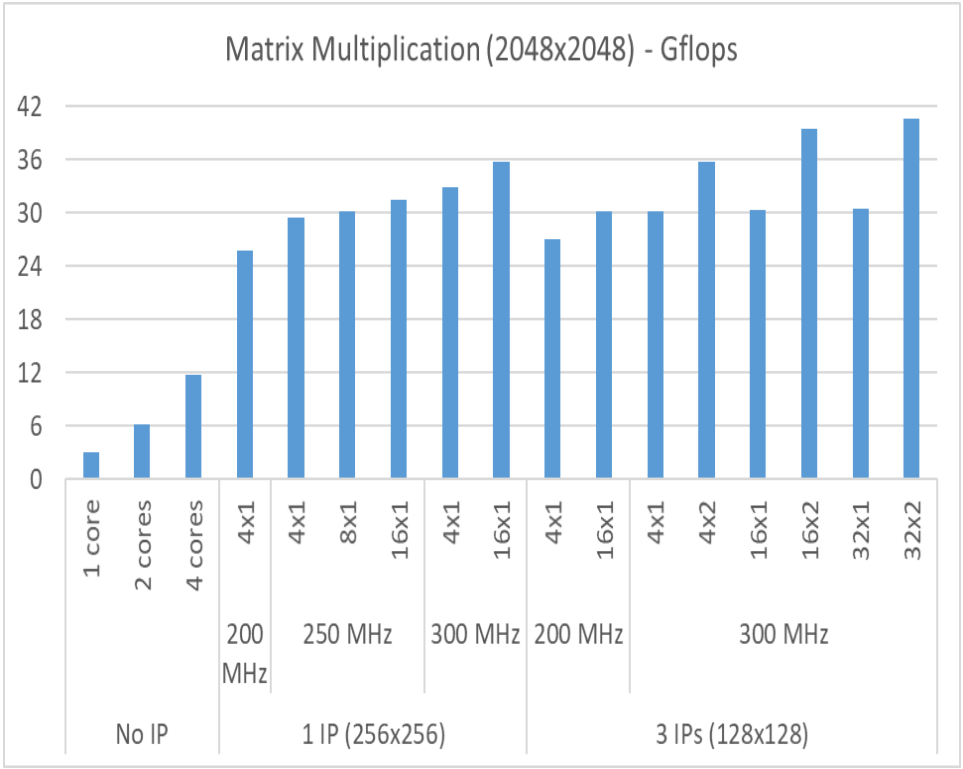
- A need if we expect programmers to survive
  - Architectures appear like mushrooms
- Productivity
  - Develop once → run everywhere
- Performance
- Key concept behind OmpSs
  - **Sequential task based** program on **single address/name space**
  - **+ directionality annotations**
  - Executed in parallel: Automatic runtime computation of dependences among tasks
  - **LEGaTO**: Extend tasks with resource requirements, propagate through the stack to find the most energy efficient solution at run time

Example for OmpSs@FPGA (Matrix multiply with OmpSs and Vivado HLS annotations)

```
#define BS 128

#pragma omp target device(fpga) copy_deps onto(0) num_instances(3)
#pragma omp task in(a,b) inout(c)
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;
    }
}

…
for (i_b=0; i_b<NB_I; i_b++)
  for (j_b=0; j_b<NB_J; j_b++)
    for (k_b=0; k_b<NB_K; k_b++)
      matrix_multiply(AA[i_b][k_b], BB[k_b][j_b], CC[i_b][j_b]);
…
```

OmpSs@FPGA Experiments (Matrix multiply)

IPs configuration	1*256, 3*128	Number of instances * size
Frequency (MHz)	200, 250, 300	Working frequency of the FPGA
Number of SMP cores	SMP: 1 to 4 FPGA: 3+1 helper, 2+2 helpers	Combination of SMP and helper threads
Number of FPGA helper threads	SMP: 0; FPGA: 1, 2	Helper threads are used to manage tasks on the FPGA
Number of pending tasks	4, 8, 16 and 32	Number of tasks sent to the IP cores before waiting for their finalization



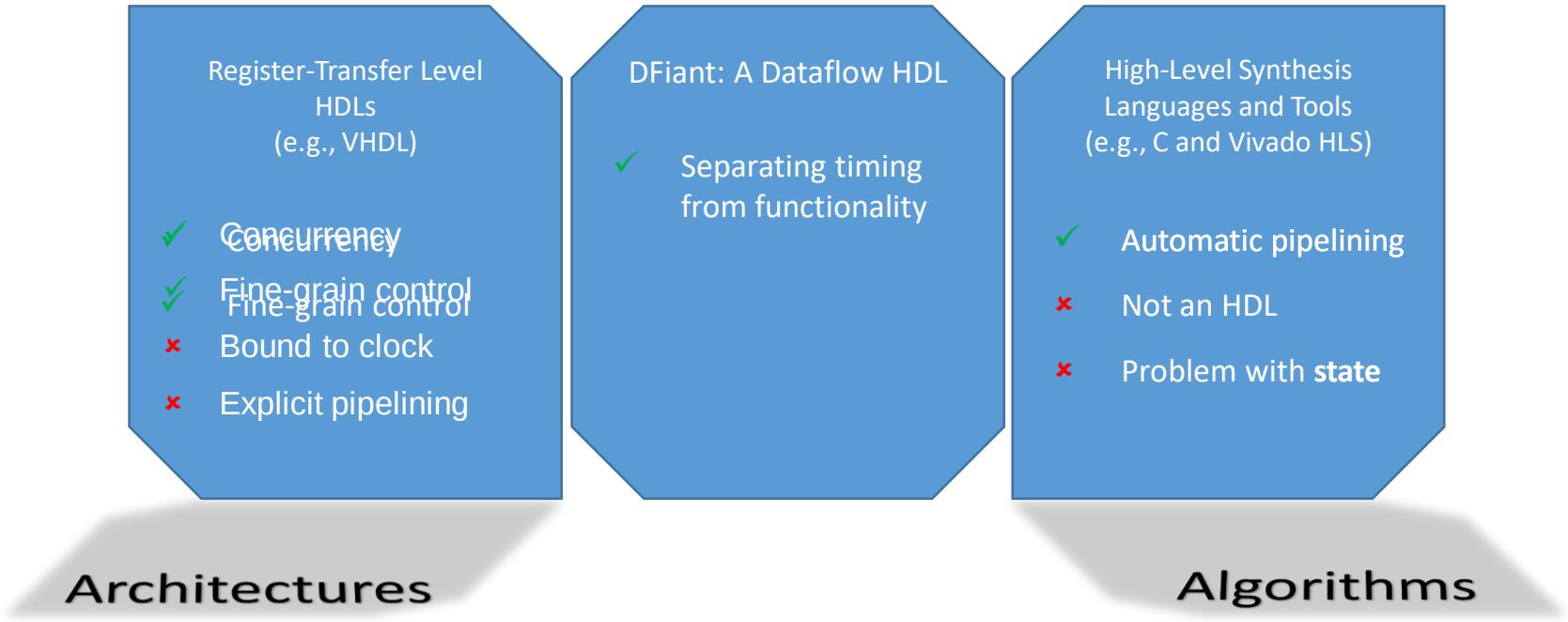
AXIOM board:
Xilinx Zynq Ultrascale+ chip, with 4 ARM Cortex-A53 cores, and the ZU9EG FPGA.

LEGaTO
www.legato-project.eu

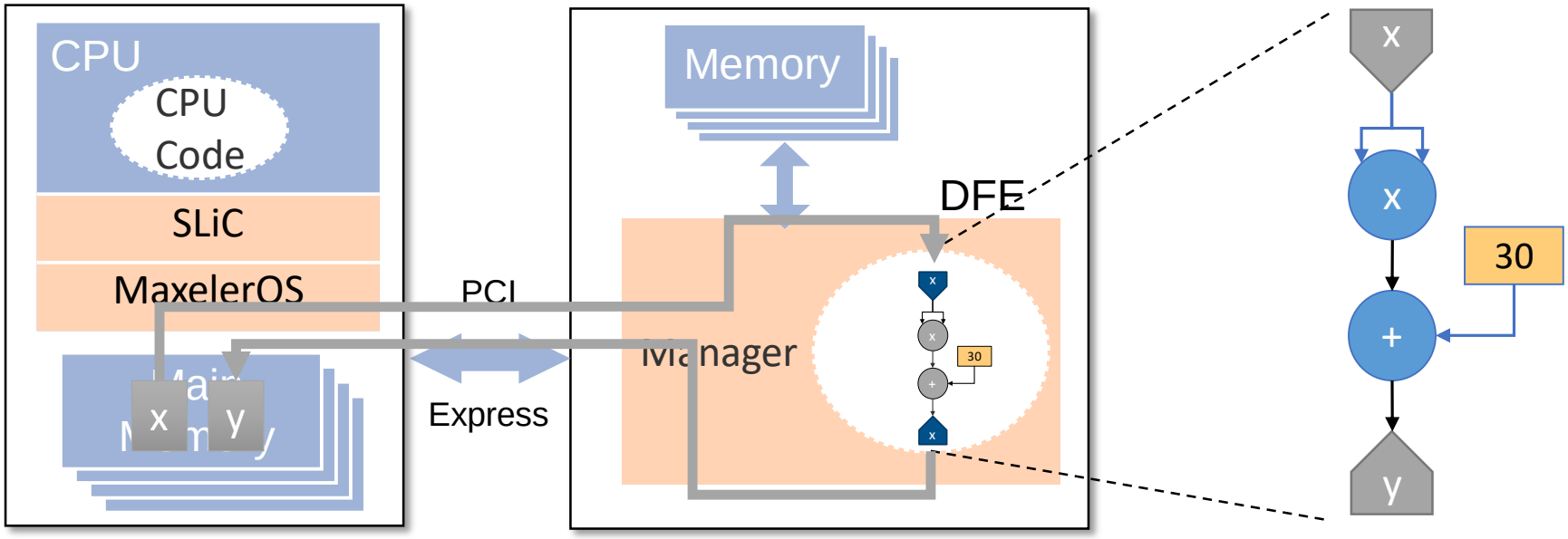
DFiant HLS

- Aims to bridge the programmability gap by combining constructs and semantics from software, hardware and dataflow languages

- Programming model accommodates a middle-ground between low-level HDL and high-level sequential programming



MaxCompiler: Application Development



Host Code (.c)

```
int*x, *y;
for (int i =0; i < DATA_SIZE; i++)
   y[i]= x[i] * x[i] + 30;
                y, DATA_SIZE*4);
```

MyManager (.maxj)

```
Manager m = new Manager();
Kernel k =
    new MyKernel();

m.setKernel(k);
m.setIO(
   link("x", CPU),
   link("y", CPU));
m.build();
```

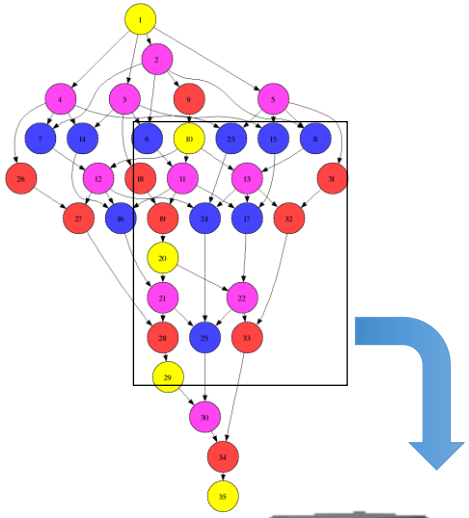
MyKernel (.maxj)

```
DFEVar x = io.input("x", dfeInt(32));

DFEVar result = x * x + 30;

io.output("y", result, dfeInt(32));
```

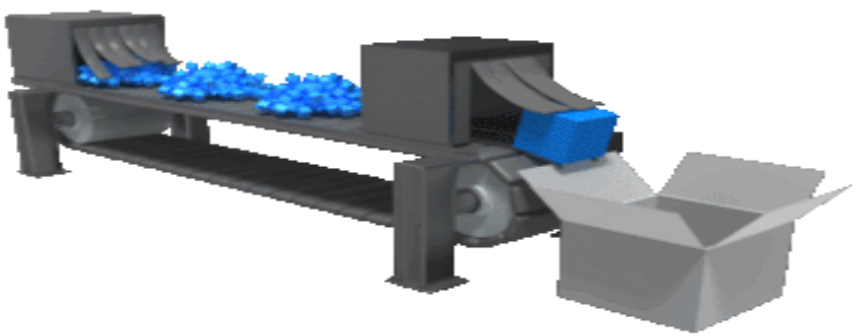
Task-based kernel identification/DFE mapping

OmpSs identifies “static” task graphs while running and DFE kilo-task instance is selected

Instantiate static, customized, ultra-deep (>1,000 stages) computing pipelines

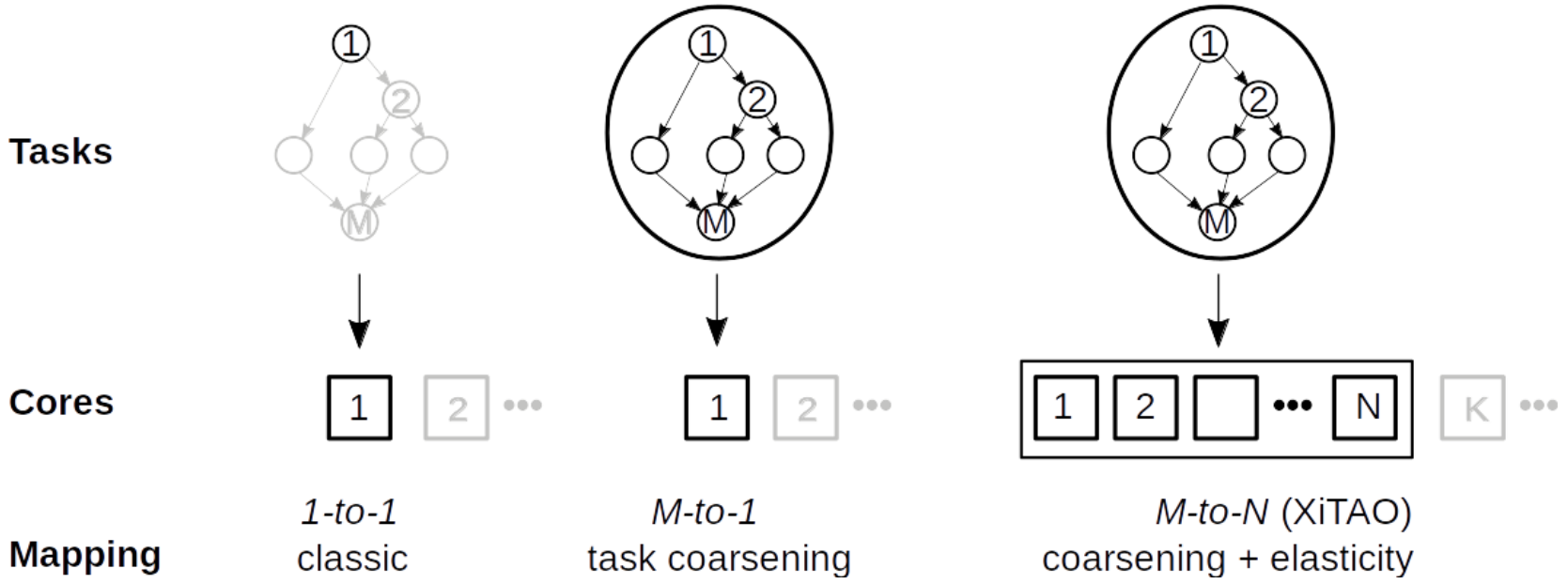


```
void Cholesky( float *A[NT][NT] ) {
int i, j, k;
for (k=0; k<NT; k++) {
   #pragma omp task inout (A[k][k])
   spotrf (A[k][k]) ;
   for (i=k+1; i<NT; i++) {
      #pragma omp task in (A[k][k]) inout (A[k][i])
      strsm (A[k][k], A[k][i]);
   }
   for (i=k+1; i<NT; i++) {
      for (j=k+1; j<i;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



DataFlow Engines (DFEs) produce one pair of results at each clock cycle

XiTAO: Generalized Tasks

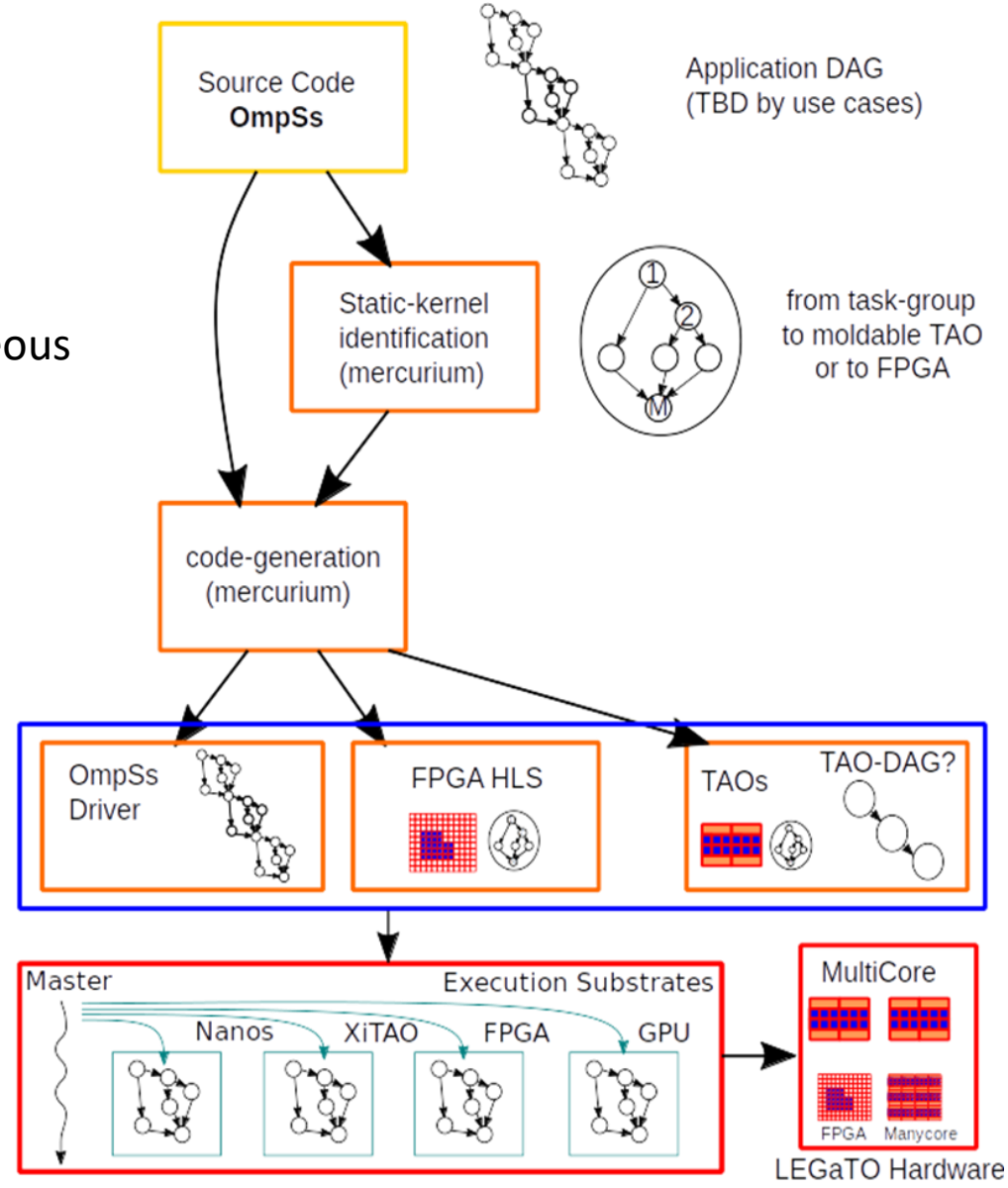


Reduces Overheads
Reduces Parallel Slackness
Overcommits memory resources

- Improve Parallel Slackness by reducing dimensionality
- Reduce Cache/BW pressure by allocating multiple resources

OmpSs + XiTao

Extend the XiTAO with support for heterogeneous resources



Why extremely low-voltage for FPGAs?

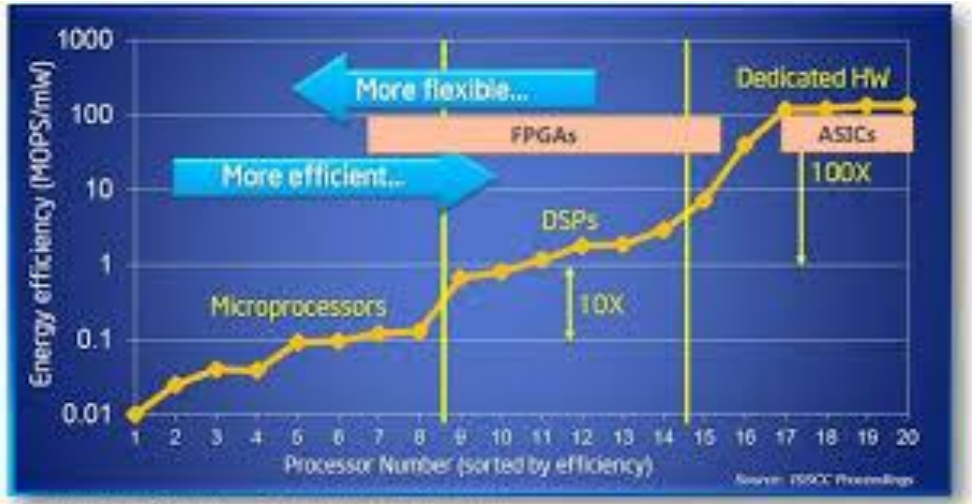
Why Energy efficiency through Undervolting?

- Still way to go for <20MW@exascale
  - FPGAs are at least 20X less power- and energy-efficient than ASICs.

- Undervolting directly delivers dynamic and static power/energy reduction.

- Note: undervolting is not DVFS
  - Frequency is not lowered!

$$P_d = \alpha C_l V_{dd}^2 F + I_0 \times 10^{-V_t/S} V_{dd}$$

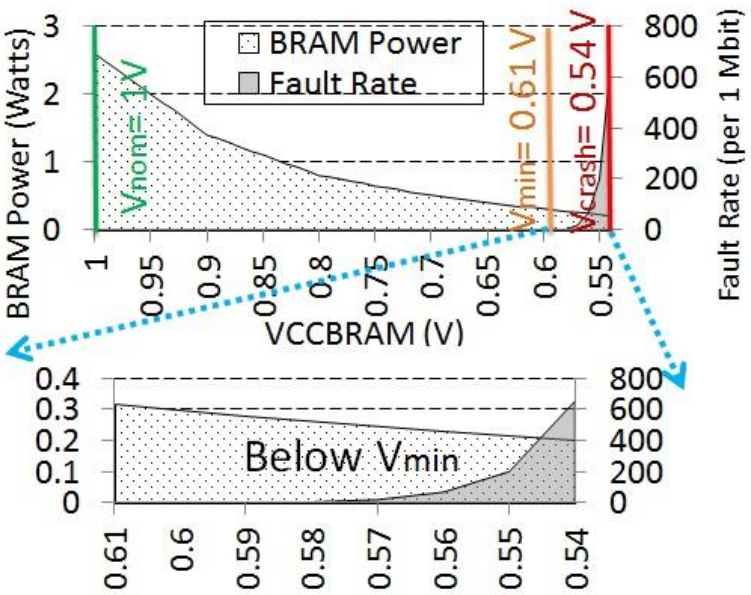


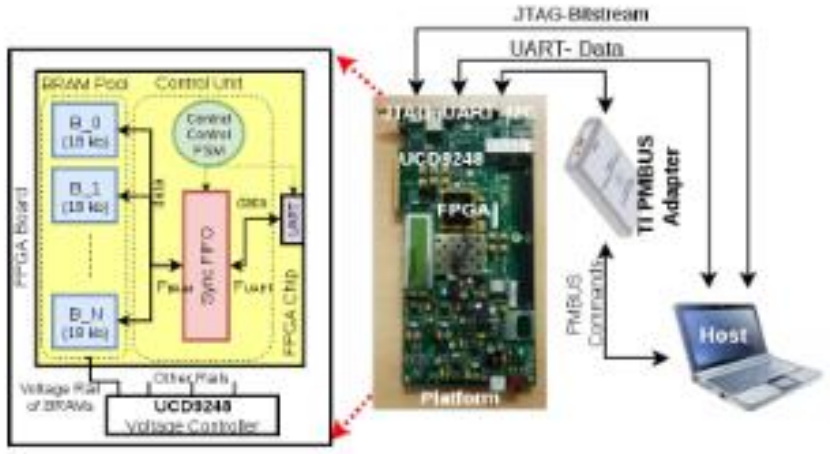
Source: Bob Brodersen, Berkeley Wireless group

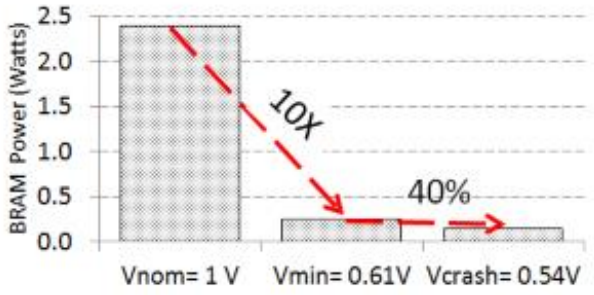
FPGA Undervolting Preliminary Experiments

Voltage scaling below standard nominal level, in on-chip memories of Xilinx technology:

- Power quadratically decreases, deliver savings of more than 10X.
- A conservative voltage guardband exists below nominal.
- Fault exponentially increases, below the guardband
- Faults can be detected / manage

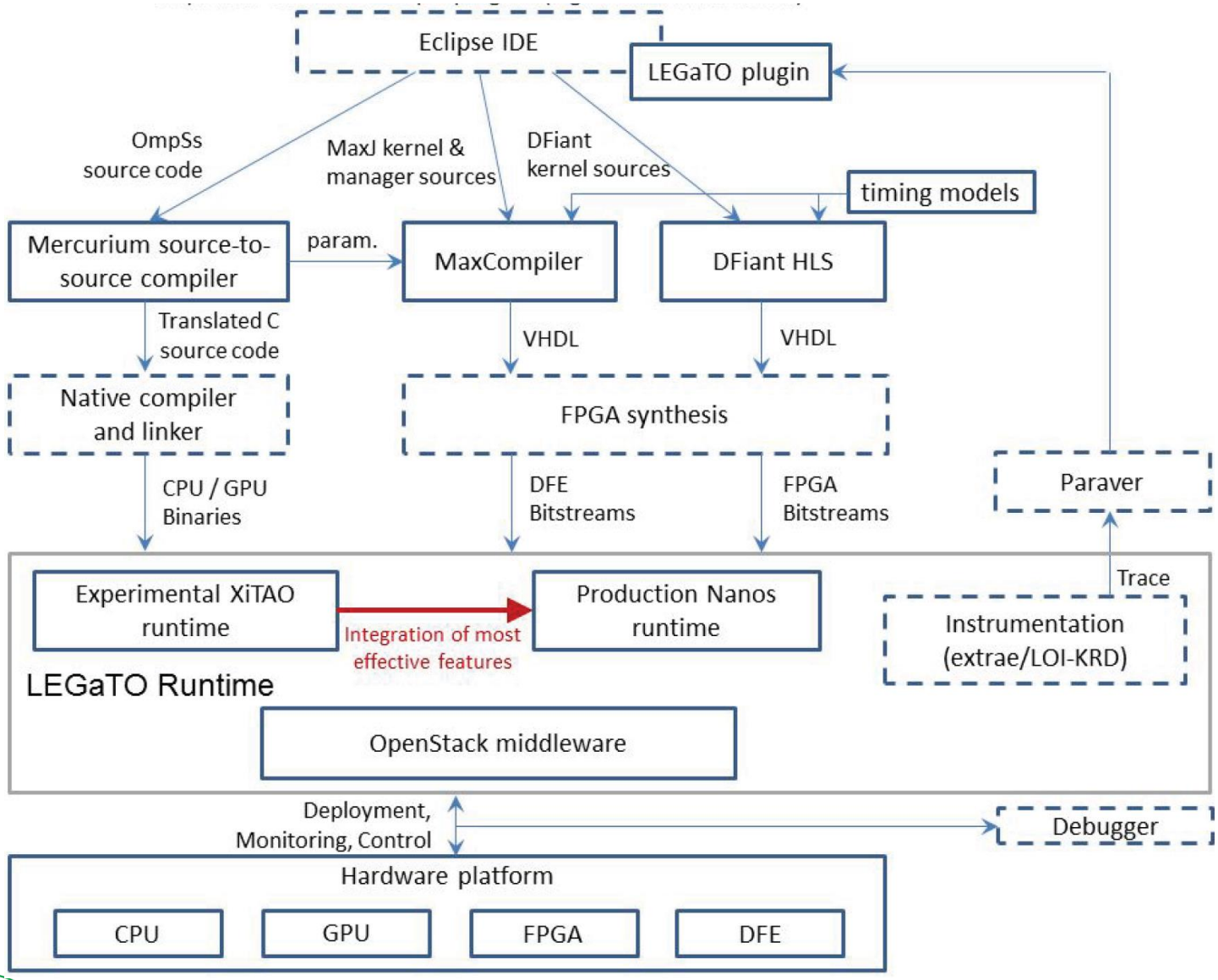






LEGaTO
www.legato-project.eu

LEGaTO System Stack




Questions?
https://legato-project.eu/
legato-project@bsc.es
LEGaTO
www.legato-project.eu