Targeting the Parallella

Spiros N. Agathos
Alexandros Papadogiannakis
Vassilios V. Dimakopoulos

Department of Computer Science & Engineering
UNIVERSITY OF IOANNINA
Ioannina, Greece

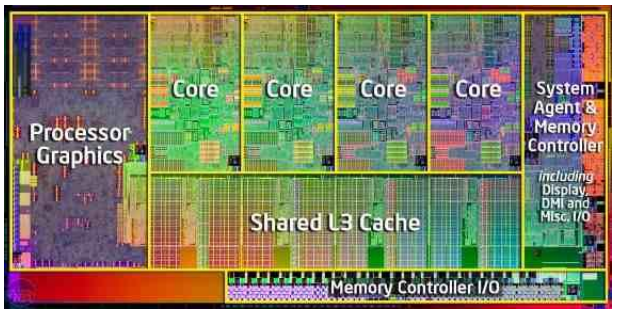
Euro-Par 2015 Vienna

IKY
ΙΔΡΥΜΑ ΚΡΑΤΙΚΩΝ ΥΠΟΤΡΟΦΙΩΝ
STATE SCHOLARSHIPS FOUNDATION

Presentation Overview

- Area of Interest
  - OpenMP Device Model
  - Parallella Board/Epiphany
- Implementing OpenMP 4.0 on the Parallella
  - Compiler Transformations
  - Runtime Architecture
  - OpenMP Within the Epiphany
- Experimental Results

Area of Interest



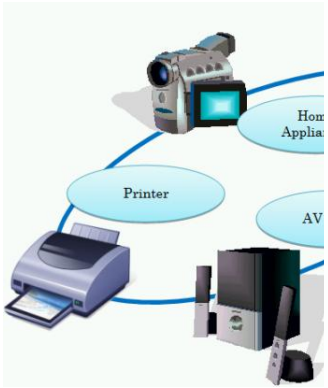
Tianhe-2 (China)

16,000 nodes X {2 Intel Xeon + 3 Xeon Phi}

RANK 1st (June 2015)

- Market trends (CPU + coprocessor):
  - Home appliances now include embedded systems
  - Personal workstations → Multiple compute cores in a socket
  - HPC supercomputers → Multicore CPUs / GPGPUs / DSPs / FPGAs
- The building blocks of heterogeneous computing nodes are designed for different workload scenarios
  - Multicore CPUs perform best in coarse grained tasks
  - Accelerators are designed for large scale data and fine grained vector processing

Area of Interest



Easy Programming?

2 (China)

3 Xeon Phi}

- Market
  - Hom
  - Pers
  - HPC
- The buil differen
  - Mult
  - Acce proc

Code Portability?

gned for

Efficient Programming?

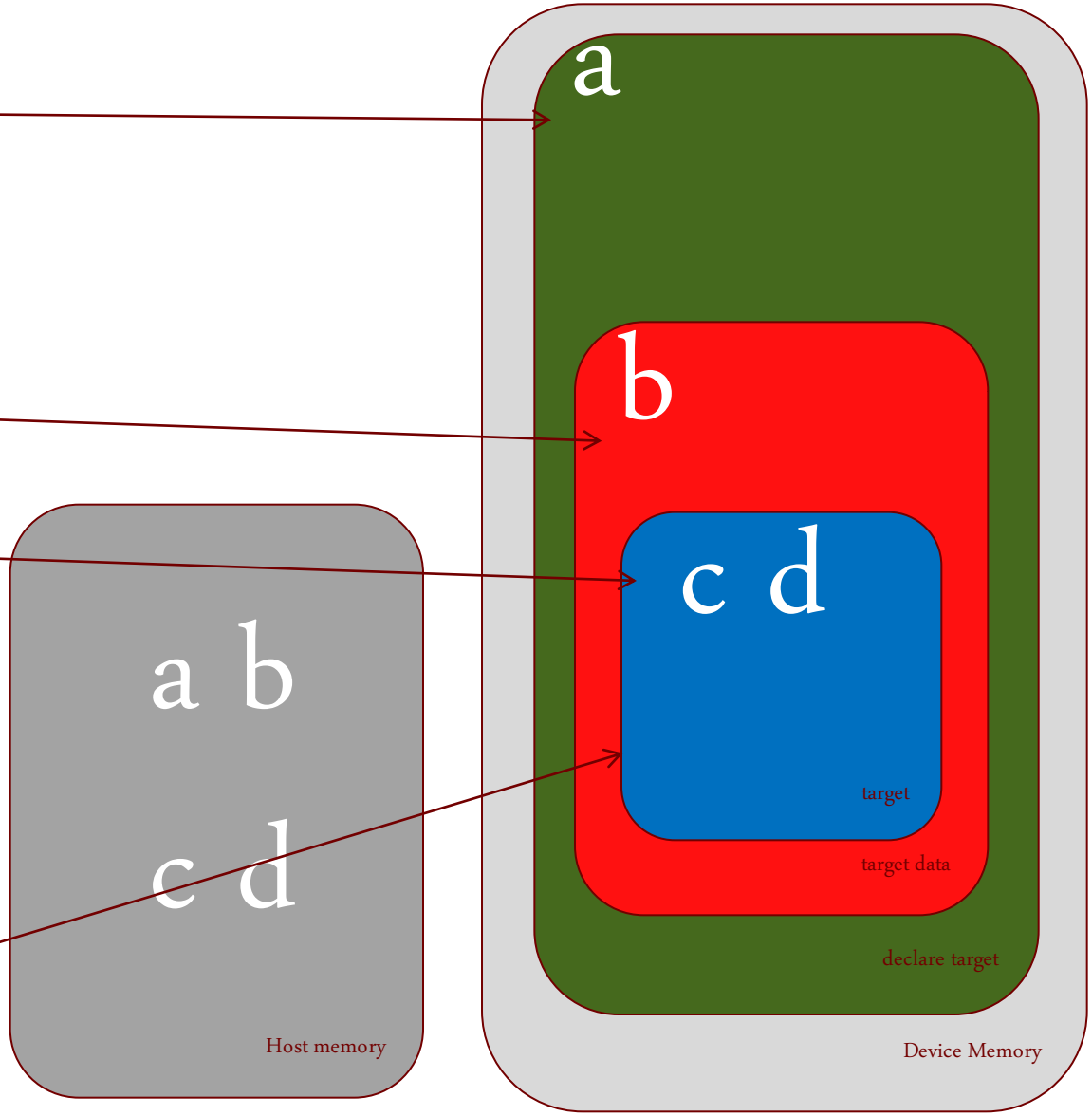
tor

OpenMP Device Model (1/2)

- OpenMP
  - Directive-based model for parallel programming
  - Base language (C/C++/Fortran)
  - Incremental parallelization
  - Highly successful, widely used, supported by many compilers
- Device Support
  - First included in the latest specification (v4.0)
  - Create data environment in the device
  - Instruct device to execute code regions (kernel)

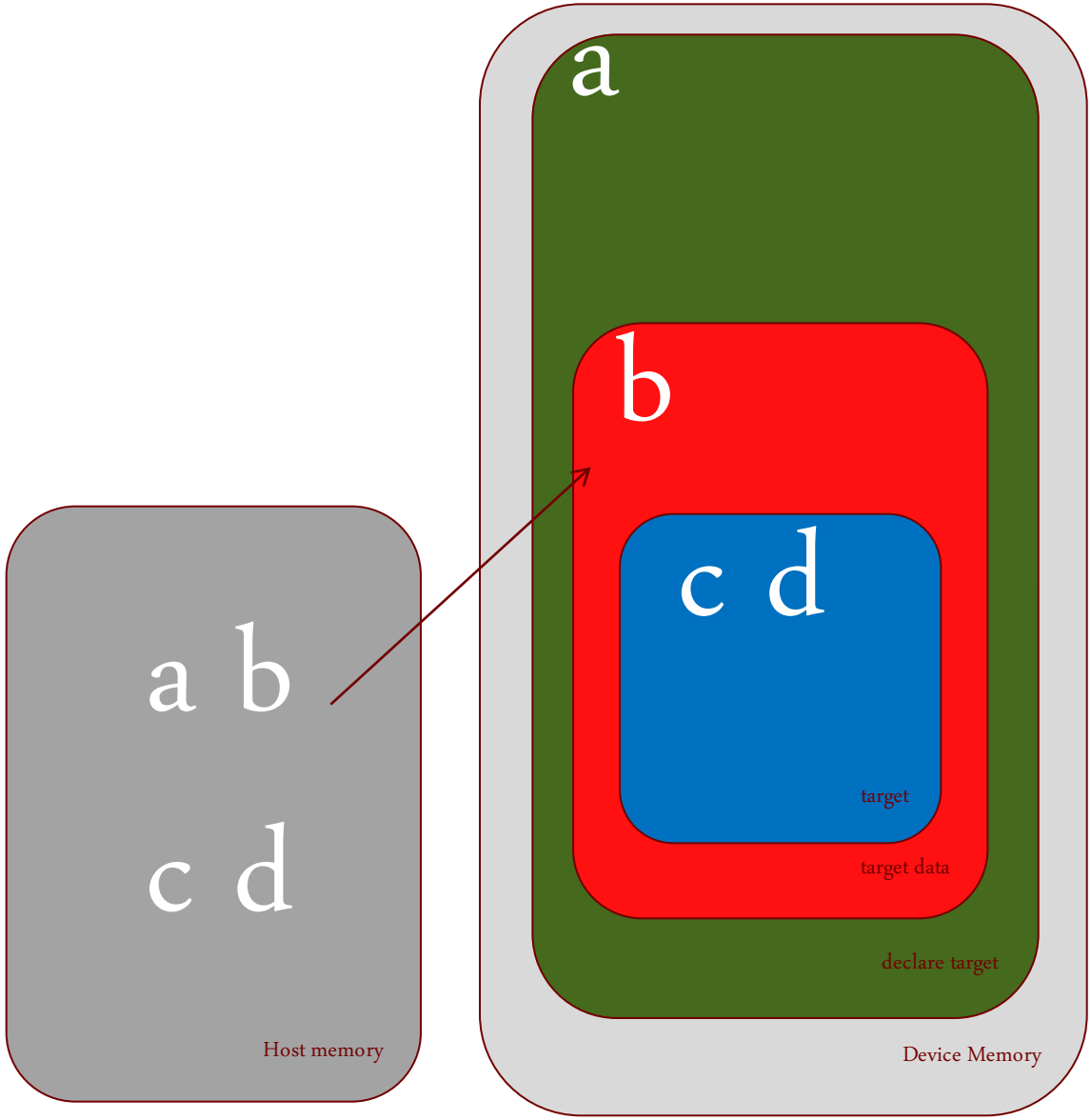
OpenMP Device Model (2/2)

```
#pragma omp declare target
  int a;
#pragma omp end declare target



#pragma omp target data map(b)
{
  #pragma omp target map(c, d)
  {
     d = a + b + c;
  }
  b += d;
  #pragma omp target update to(b)
  #pragma omp target {…}
}
```

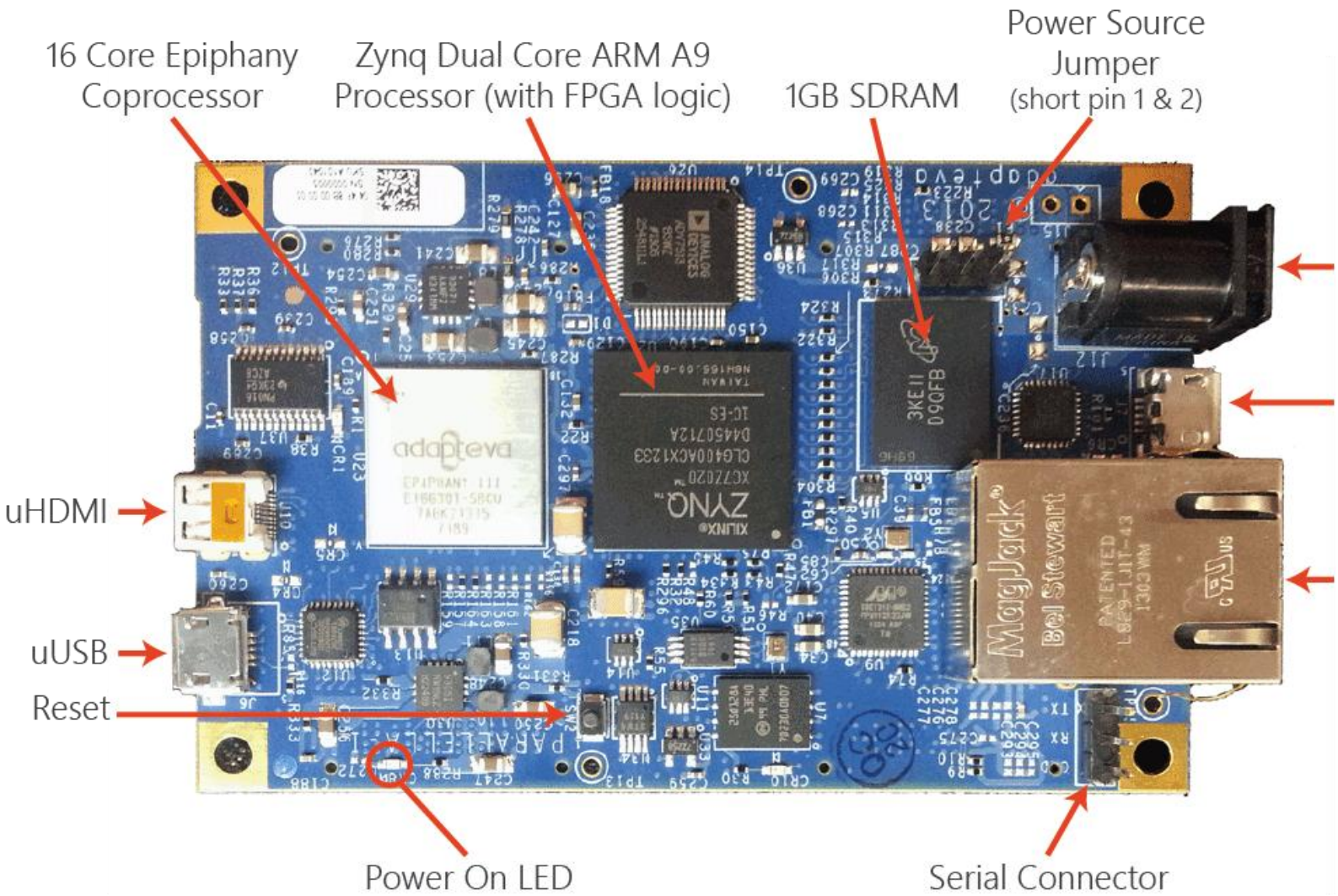


OpenMP Device Model (2/2)

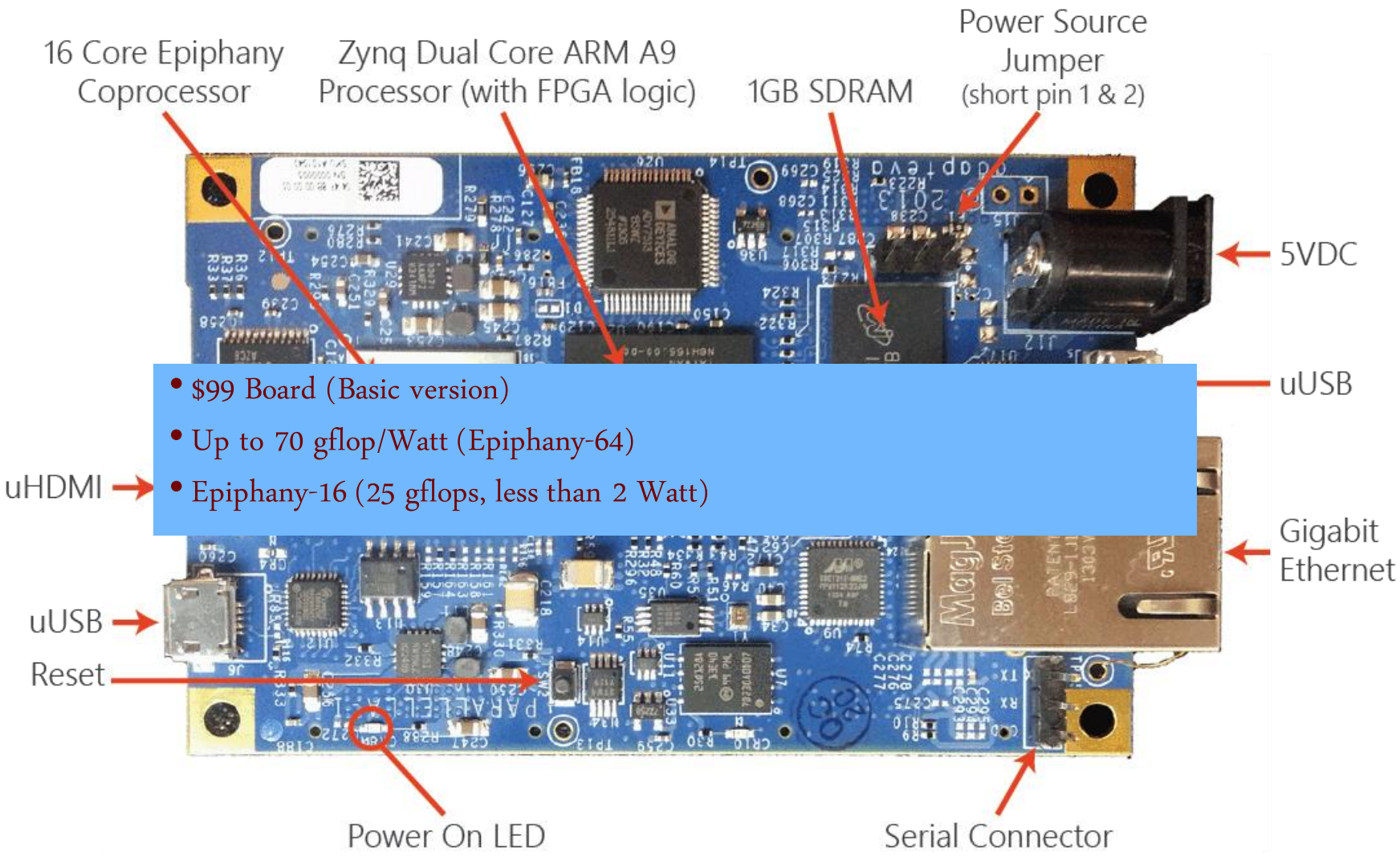
```
#pragma omp declare target
  int a;
#pragma omp end declare target



#pragma omp target data map(b)
{
  #pragma omp target map(c, d)
  {
     d = a + b + c;
  }
  b += d;
  #pragma omp target update to(b)
  #pragma omp target {…}
}
```



Parallella Board

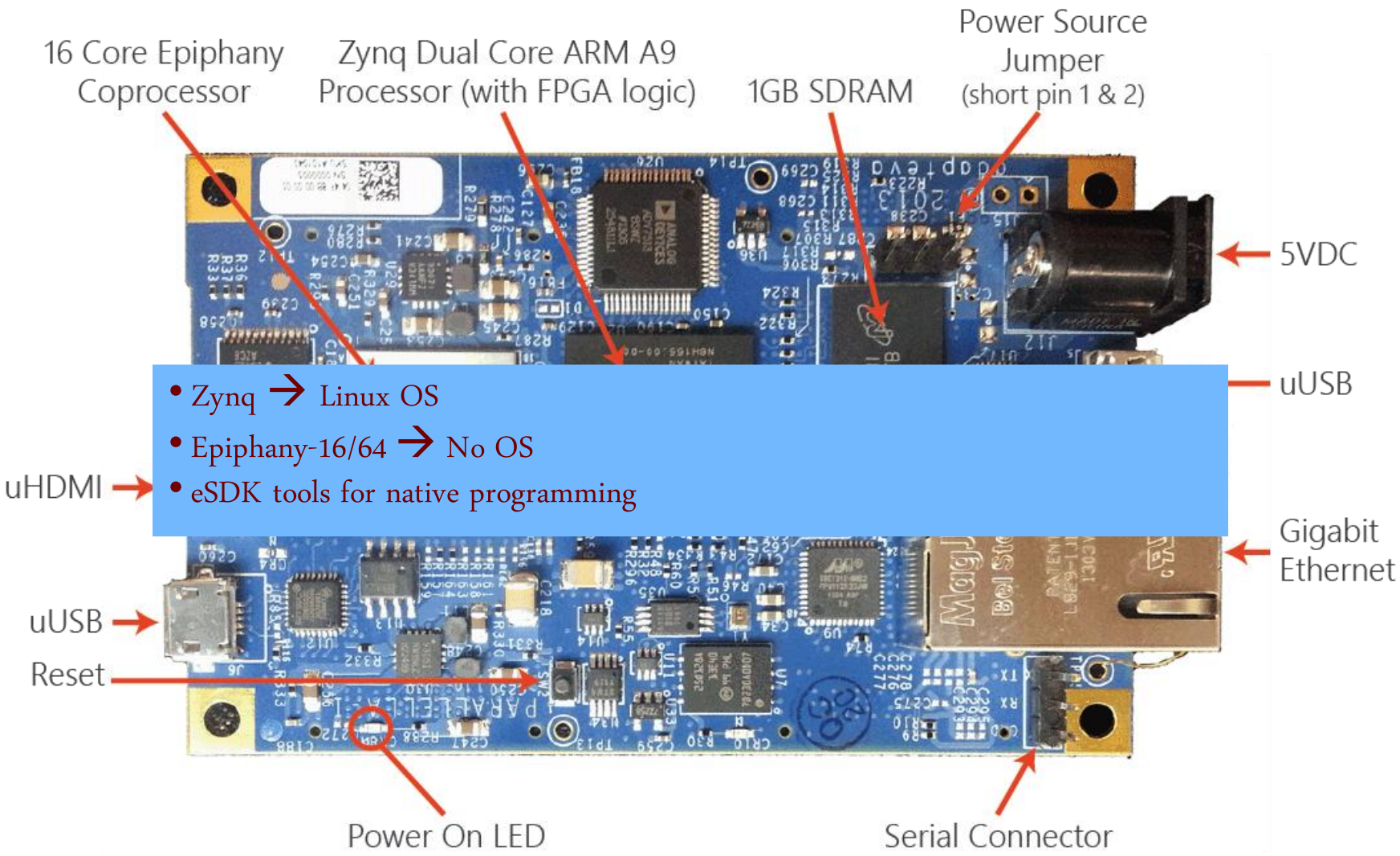


Parallella Board



- $99 Board (Basic version)
- Up to 70 gflop/Watt (Epiphany-64)
- Epiphany-16 (25 gflops, less than 2 Watt)

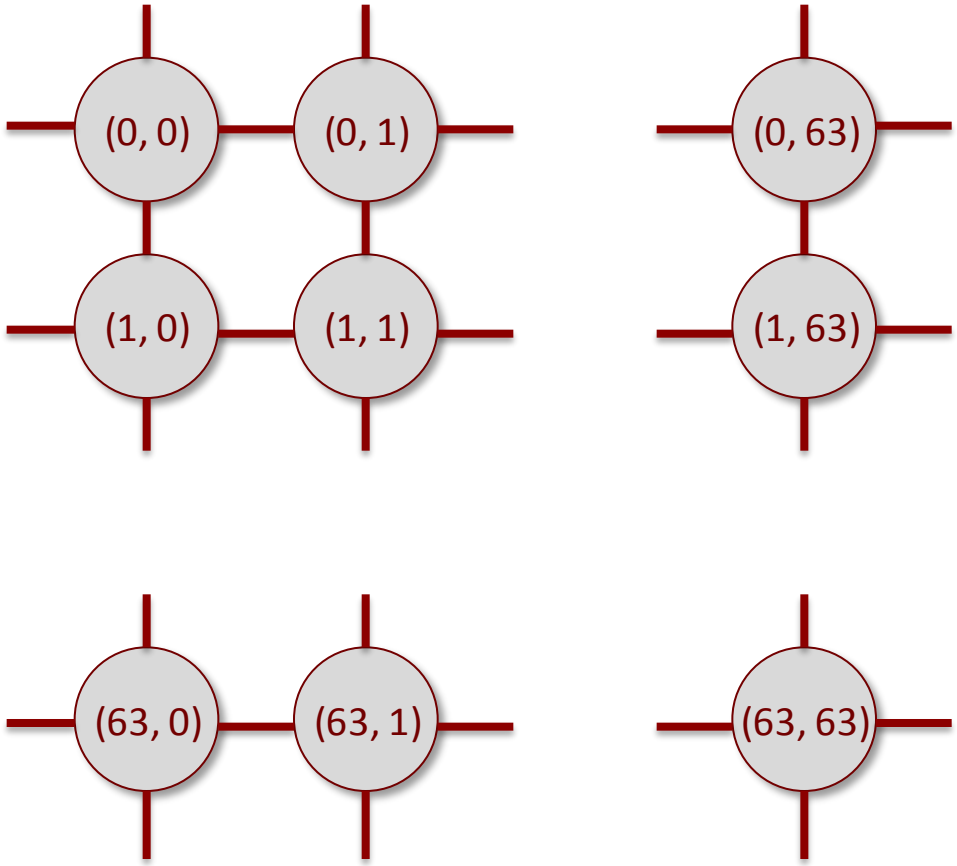
Parallella Board



- Zynq → Linux OS
- Epiphany-16/64 → No OS
- eSDK tools for native programming

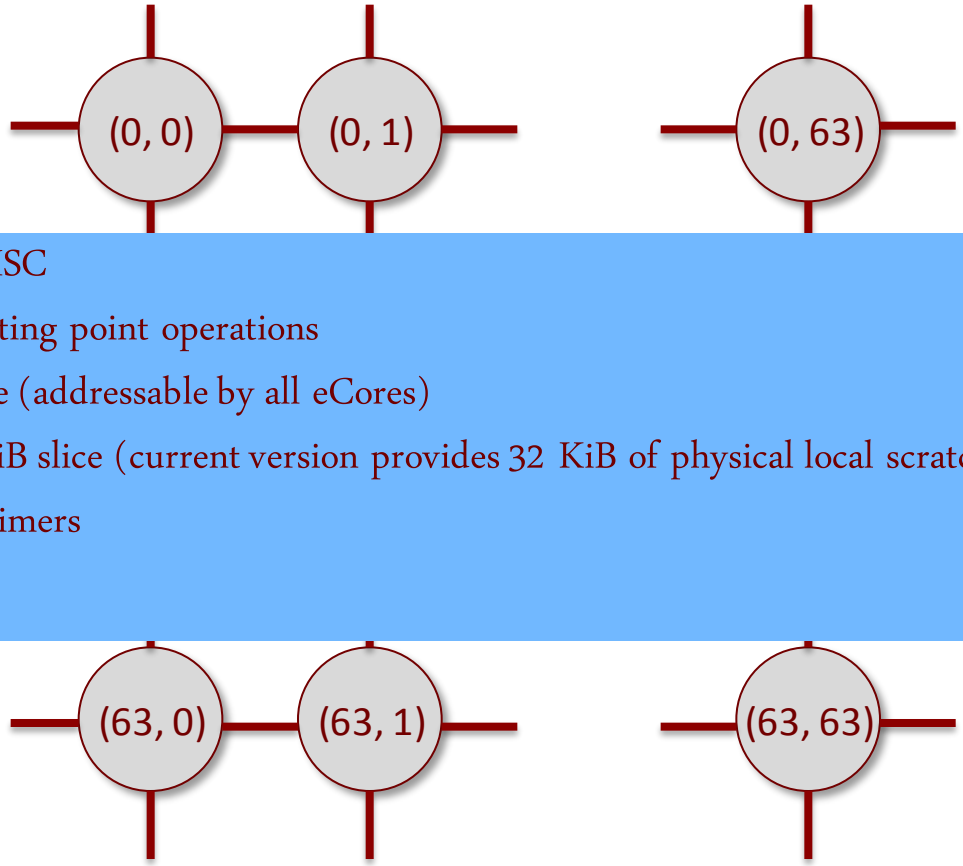
Epiphany Coprocessor (1/2)

- 64x64 mesh => 4096 eCores maximum



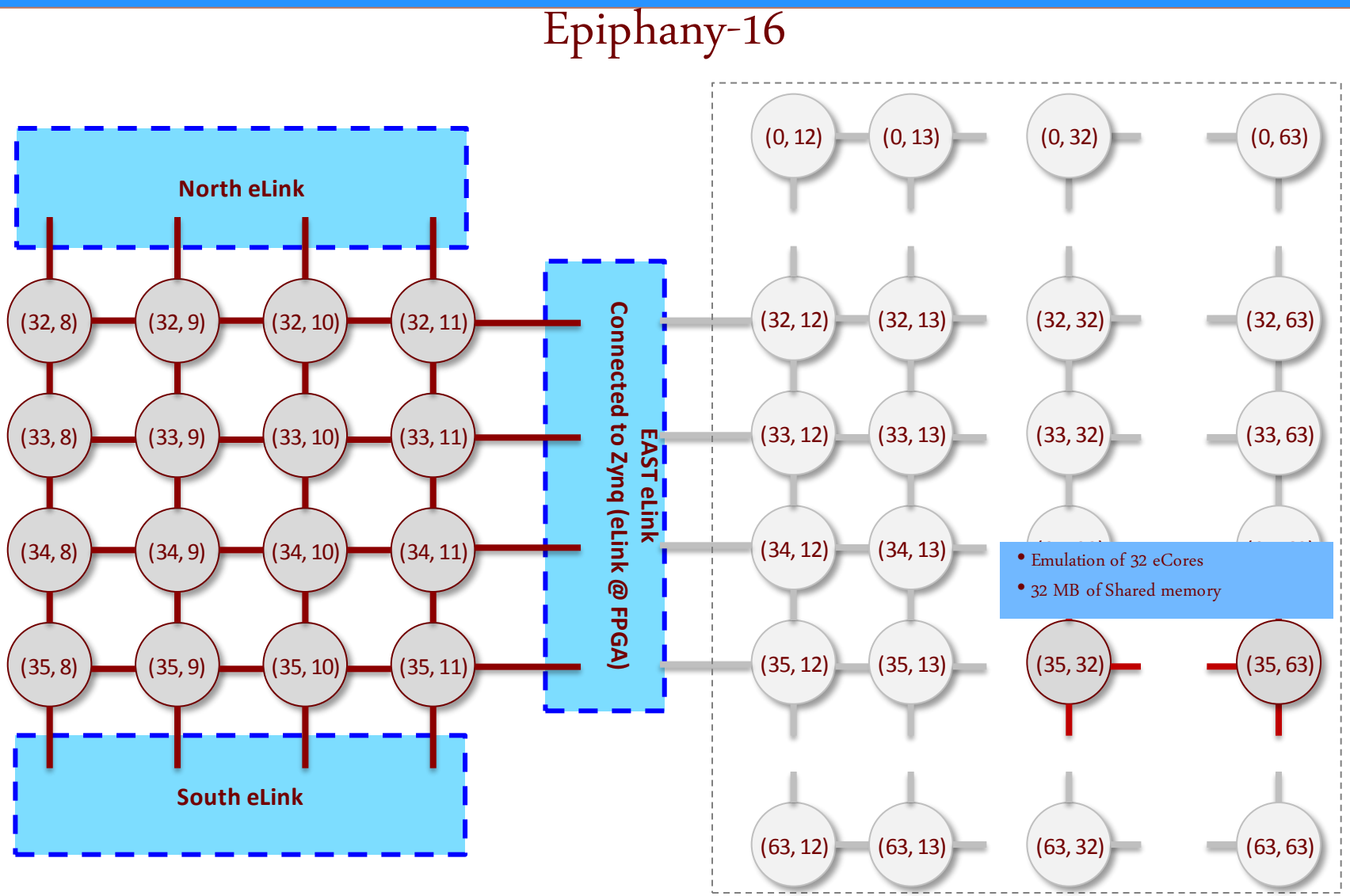
Epiphany Coprocessor (1/2)

- 64x64 mesh => 4096 eCores maximum



- 32-bit superscalar RISC
- Single-precision floating point operations
- Global address space (addressable by all eCores)
- Each core owns 1MiB slice (current version provides 32 KiB of physical local scratchpad memory)
- 2 DMA engines, 2 timers

Epiphany Coprocessor (2/2)

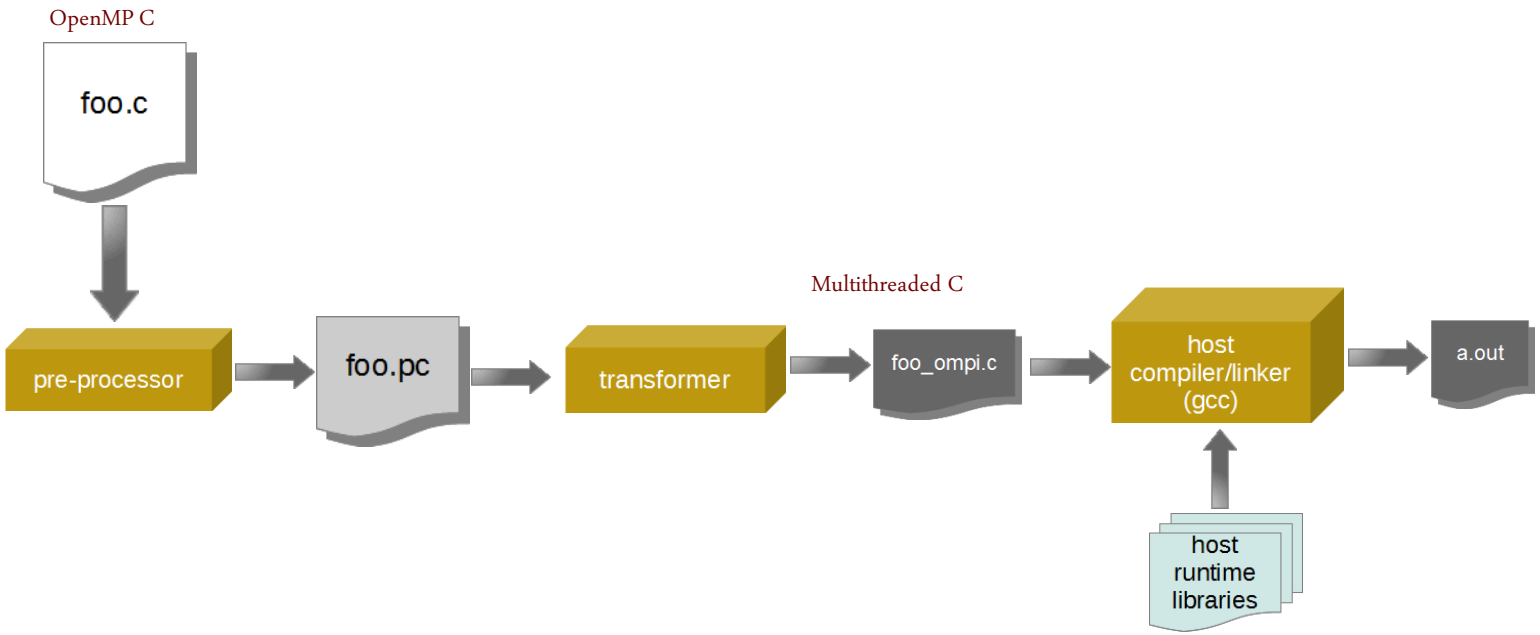


Implementing OpenMP 4.0 on the Parallella

Implementing OpenMP 4.0 on the Parallella (1/8)

Compiling for the New Device Directives

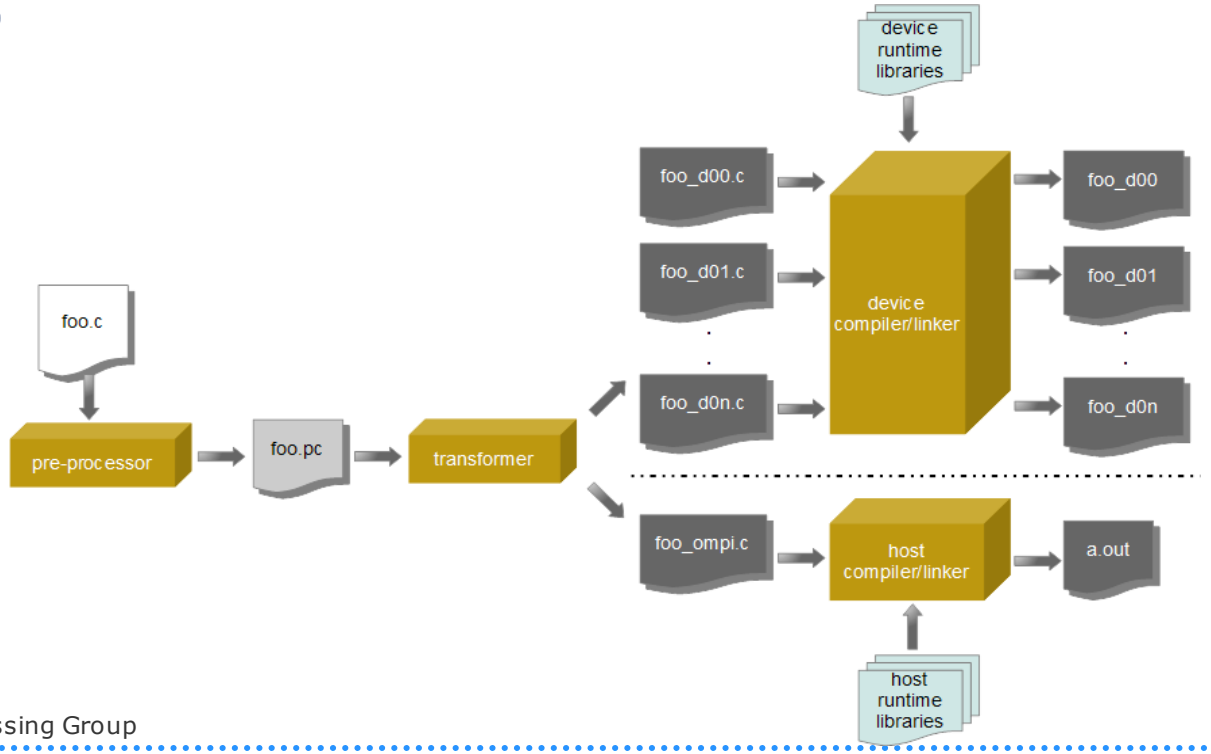
- The 1st implementation of OpenMP for the Parallella board
- OMPi (http://paragroup.cs.uoi.gr/wpsite/software/ompi):
  - V3.1 OpenMP C infrastructure
  - Source to source compiler + Runtime libraries



Implementing OpenMP 4.0 on the Parallella (2/8)

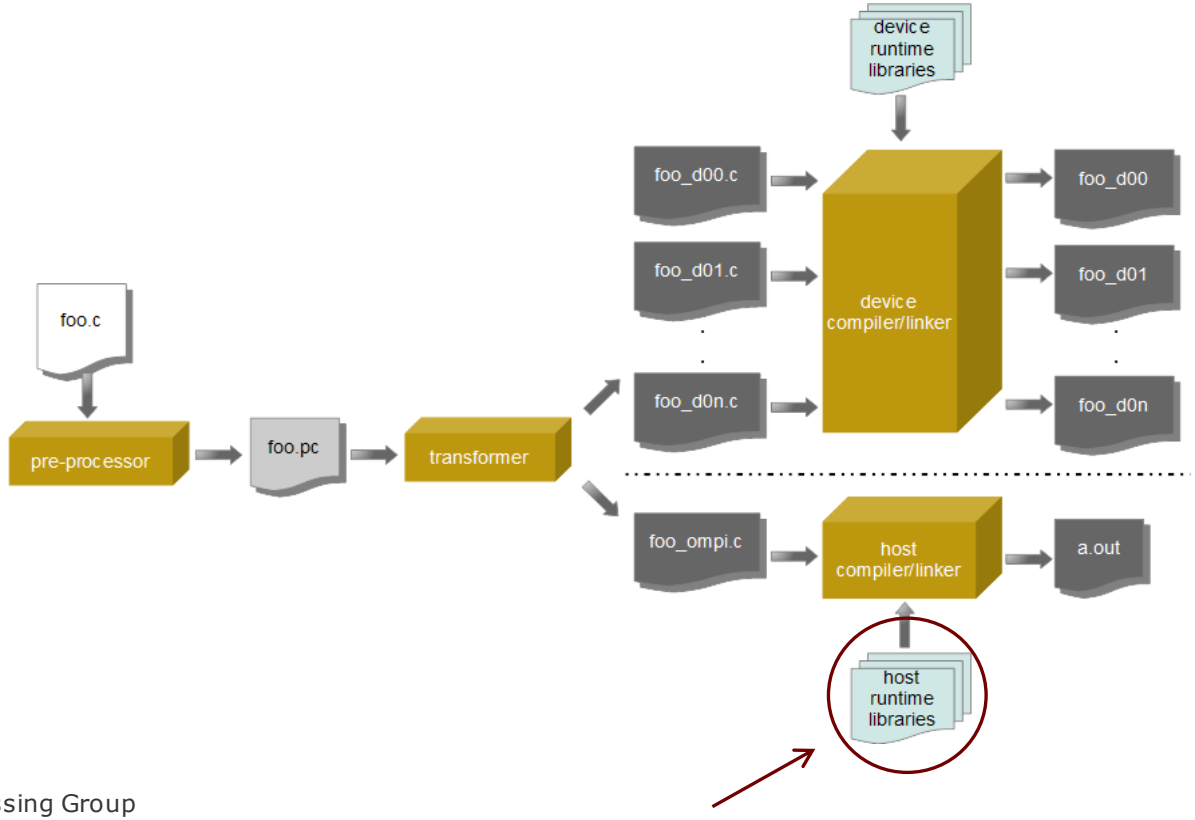
Compiling for the New Device Directives

- Input grammar modified with the new target-related directives
- AST augmented with new nodes to represent the new directives
- Each kernel (i.e. target region), is outlined to a separate function
- The code generation phase produces multiple output files, one for each different kernel, plus the host code (host may be called to execute any of them)



Runtime Architecture - What the Host does

- A full-fledged OpenMP runtime library
  - Supporting execution on the dual-ARM processor
- Additional functionality
  - Required for controlling and accessing the Epiphany device



Implementing OpenMP 4.0 on the Parallella (3/8)

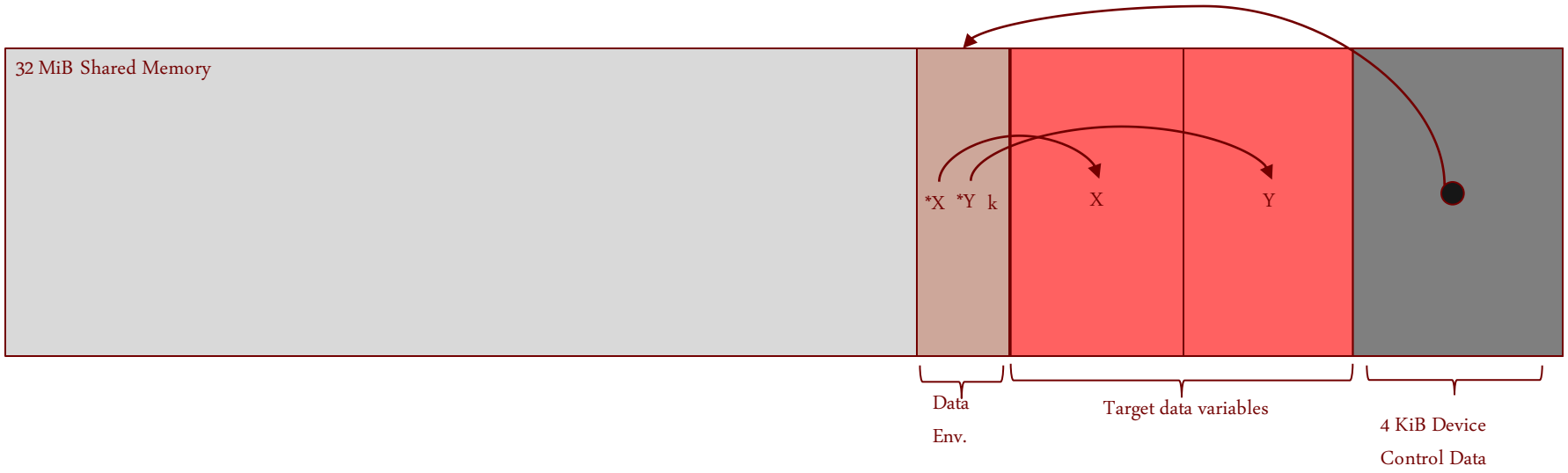
Runtime Architecture - What the Host does

- A full-fledged OpenMP runtime library
  - Supporting execution on the dual-ARM processor
- Additional functionality
  - Required for controlling and accessing the Epiphany device
- The communication between the Host and the eCores takes place through the shared memory portion of the system RAM
- Initialization phase
  - Initialize bookkeeping for each Ecore
  - Puts them to the idle state
- For offloading a kernel
  - The first idle eCore is chosen
  - Precompiled object file is loaded to it for immediate execution
  - Ecores inform Host about the completion of a kernel through special flags in shared memory.
  - Multiple host threads can offload multiple independent kernels concurrently onto the Epiphany

Implementing OpenMP 4.0 on the Parallella (4/8)

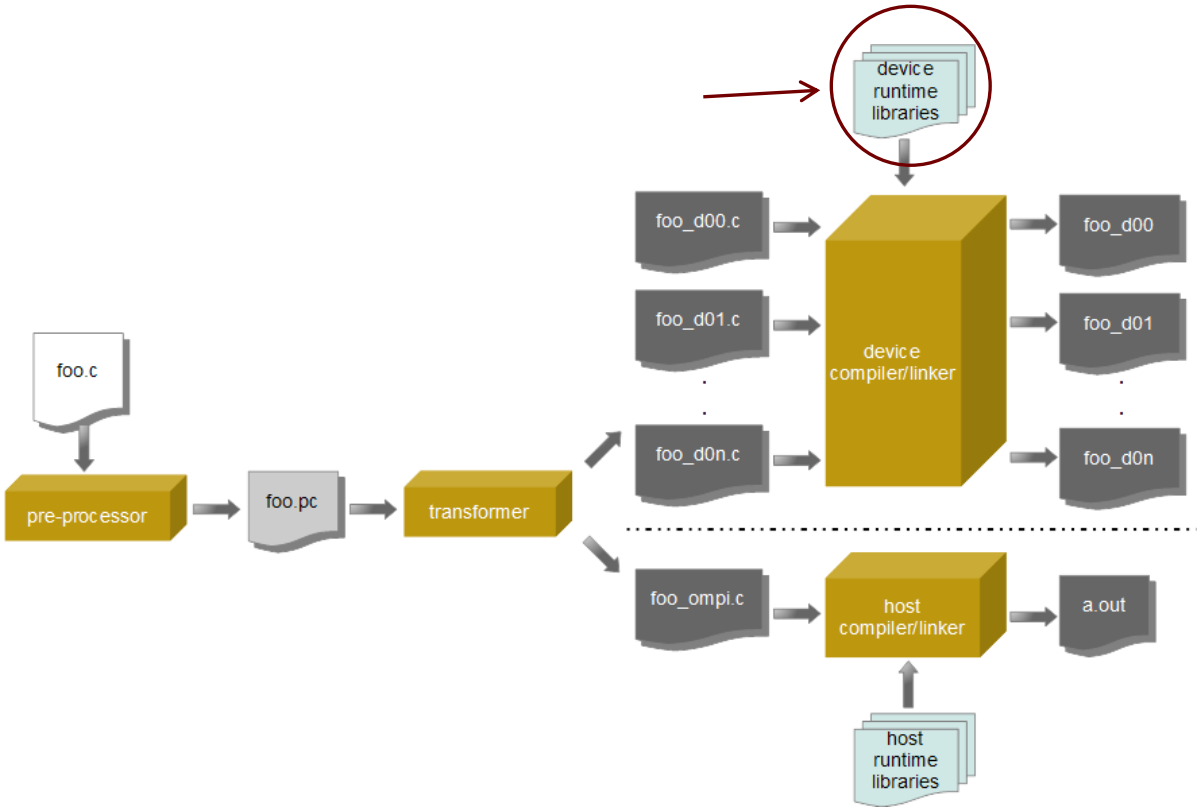
Runtime Architecture – What the Host does

```
int X[10], Y[10];
int k;

#pragma omp target data map(X,Y)
{
  #pragma omp target map(to:k)
  {
     /* Kernel code */
  }
}
```



Implementing OpenMP 4.0 on the Parallella (5/8)

OpenMP Within the Epiphany



Implementing OpenMP 4.0 on the Parallella (5/8)

OpenMP Within the Epiphany

- Supporting OpenMP within the Epiphany is nontrivial
  - ECores do not execute any OS
  - No provision for dynamic parallelism
  - The 32KiB local memory is quite limited:
    - Unable to handle sophisticated OpenMP runtime structures
- The runtime infrastructure originally designed for the Host was trimmed down to a minimum
  - This is linked and offloaded with each kernel
- The corresponding coordination among the participating eCores utilizes the local memory of the team's master eCore
  - This is possible because an ecore can access any address in the Epiphany address space
  - The mesh coordinates of the master core are available to all team eCores through the DCD area in shared memory

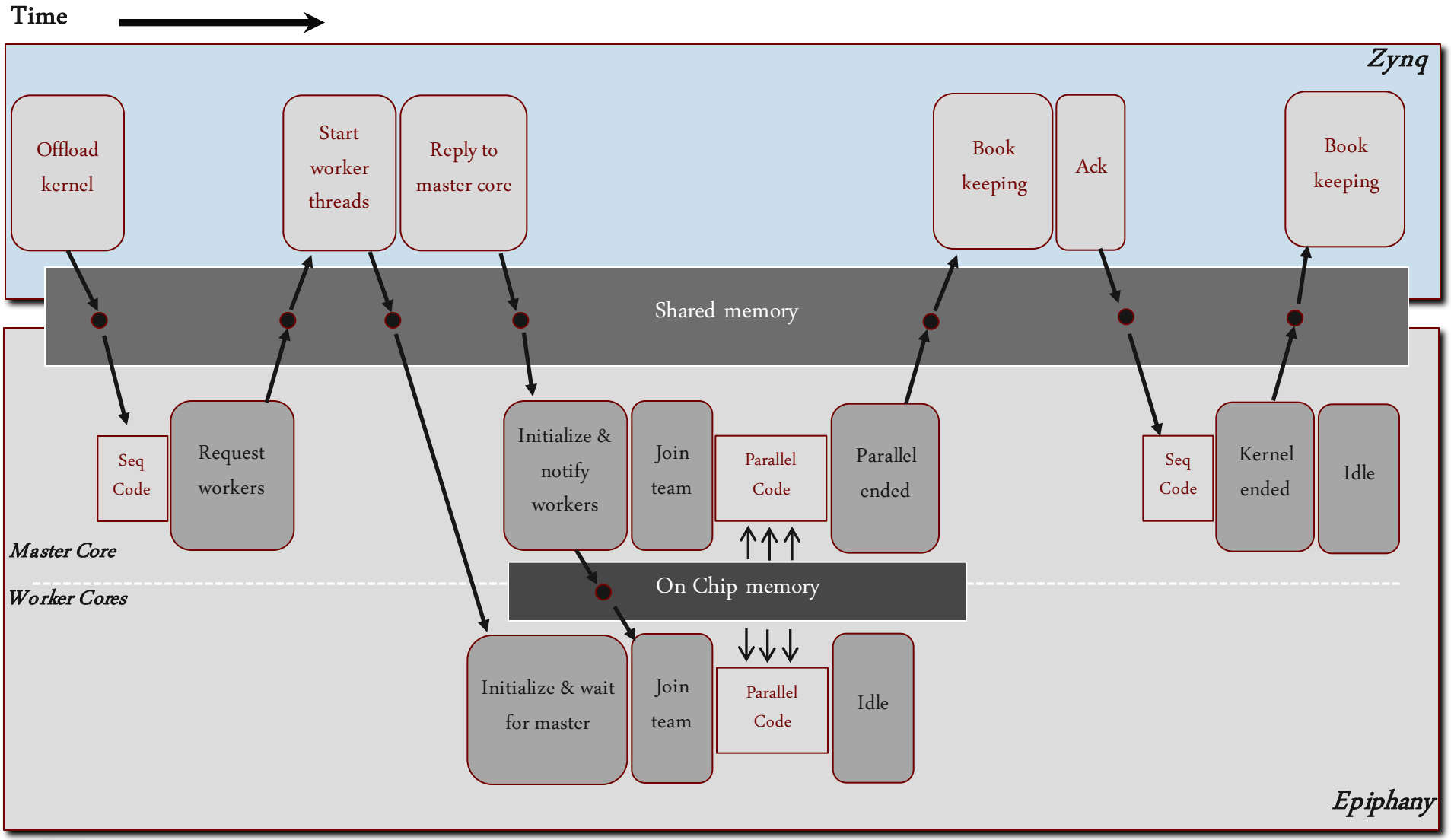
Implementing OpenMP 4.0 on the Parallella (6/8)

OpenMP Within the Epiphany

- OpenMP device model allows directives within kernel code
  - Dynamic creation of a team of threads within Epiphany is necessary
- eSDK tools:
  - Provide mechanisms to activate and control eCores only from Host side
  - Cannot be directly utilized by an OpenMP implementation
- We designed new Host assisted mechanisms to override these limitations
- Creation of parallel team…

Implementing OpenMP 4.0 on the Parallella (7/8)

#parallel inside a kernel



Implementing OpenMP 4.0 on the Parallella (8/8)

- Locks and barriers
  - Basic mechanisms for locks and barriers are provided through the eSDK libraries eCores
  - However they are not directly usable by an OpenMP runtime
  - We create custom locks and barriers that can support tasking extensions
- Our prototype tasking infrastructure is based on a blocking shared queue stored in the local memory of the master eCore
  - The corresponding task data environments are stored in the shared memory

Experimental Results

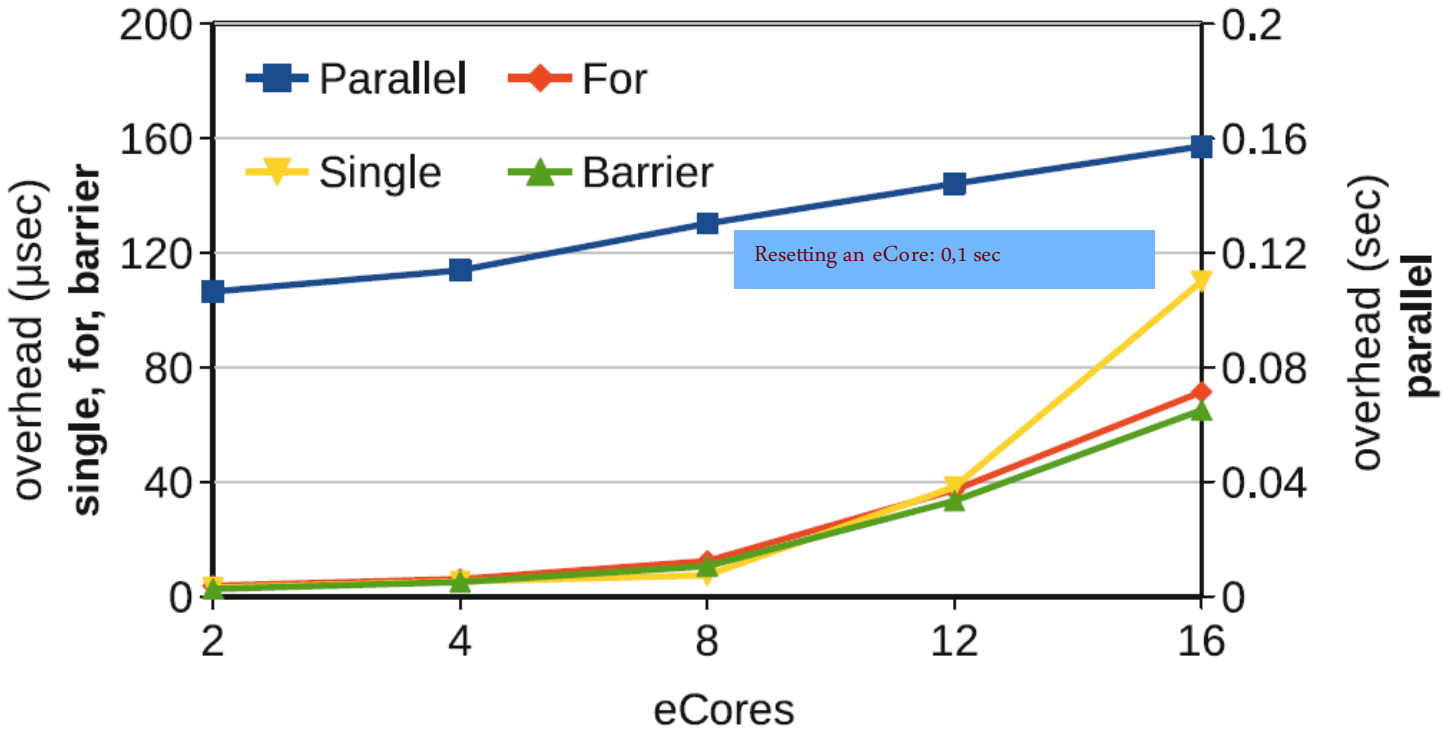
Experimental Results (1/3)

- Environment
  - Parallella-16 SKUA101020
  - Ubuntu 14.04, kernel 3.12.0 armv7l
  - gcc and e-gcc v.4.8.2 as back-end for OMPi
  - eSDK 5.13.9.10

Experimental Results (2/3)

❖ Modified version of the EPCC benchmarks

- ➢ Basic routines are offloaded through target directives
- ➢ Measurements from the host side after subtracting any offloading costs



Overhead results of EPCC benchmark

Experimental Results (3/3)

Frames per second for the Mandelbrot deep zoom application (1024x768)

#frames	eSDK@Epiphany	OMPi@Epiphany	OMPi@Zynq
204	17.854	15.829	4.139
408	15.250	13.630	3.469
612	13.411	12.292	3.015
816	12.528	11.632	2.794
1020	13.330	12.304	2.997
1224	14.486	13.234	3.288

eSDK version: only 8%-13% better

Original code: 301 lines (3 files)

OpenMP code: 198 lines (1 file)

END...

Acknowledgements:

Euro-Par 2015 Vienna

adapteva

IKY
ΙΔΡΥΜΑ ΚΡΑΤΙΚΩΝ ΥΠΟΤΡΟΦΙΩΝ
STATE SCHOLARSHIPS FOUNDATION

Thank you