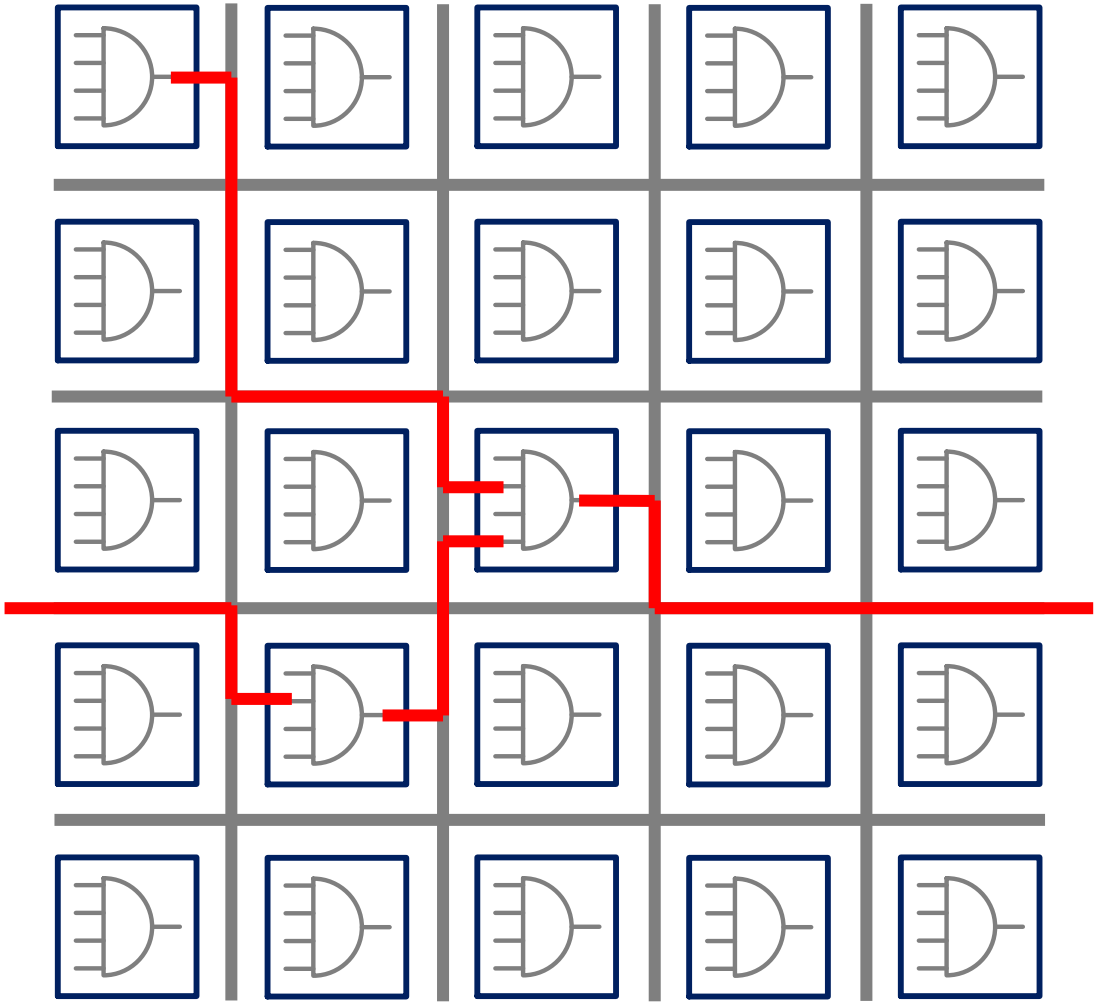
A Heterogeneous GASNet Implementation for FPGA-accelerated Computing

Ruediger Willenberg and Paul Chow

High-Performance Reconfigurable Computing Group
University of Toronto

October 9, 2014

FIELD-PROGRAMMABLE GATE ARRAYS

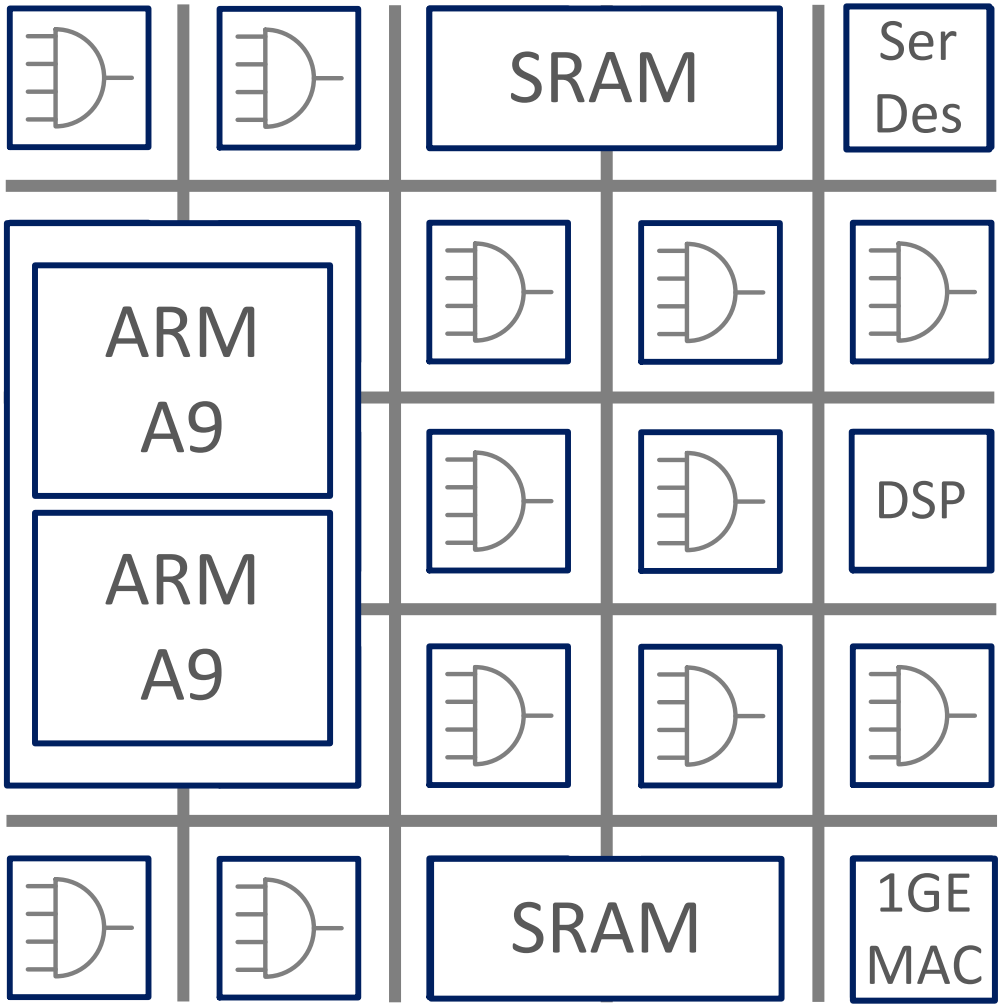


Field-Programmable Gate Arrays

© H.Roesner

© flickr / MadPhysicist

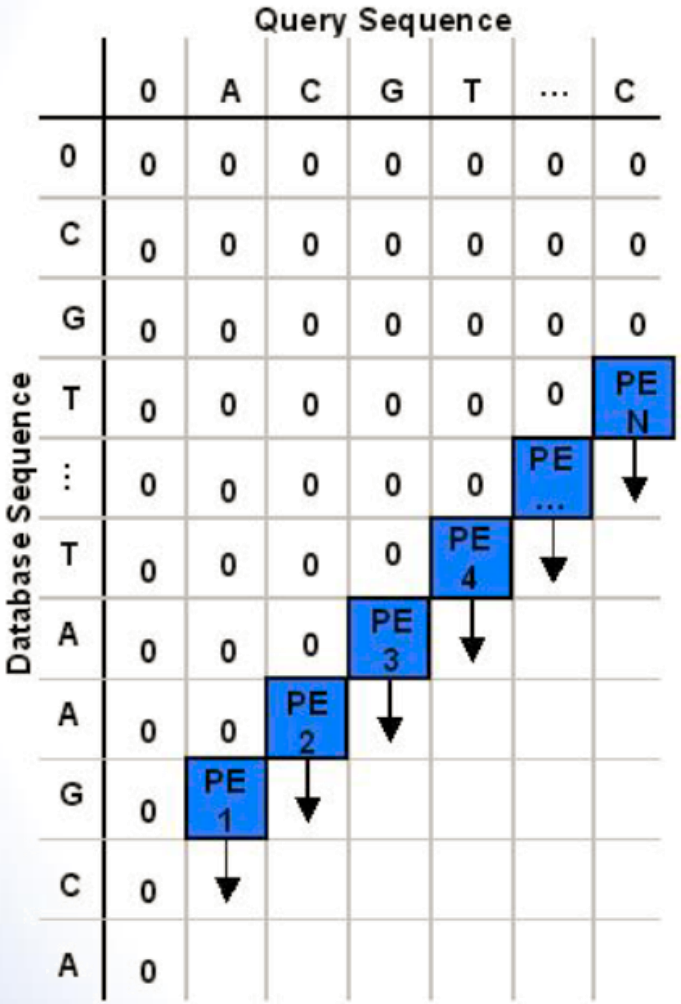
Hardwired FPGA functions



Computing with FPGAs

- Fully customized dataflow and buffering
- Tightly coupled pipelining of computations
- Very low energy / computation ratio
- Computation cores can be switched out in **msecs** with *partial reconfiguration*

Example: Smith-Waterman DNA sequencing (Dynamic Programming)



49x – 980x speedup (I/O dependent) on Xilinx V4-LX160 FPGA vs. 2.2GHz AMD Opteron

(Storaasli/Cray 2009)

FPGA programmability drawbacks

- Very different thinking than software programming
- Established Hardware Description Languages (Verilog HDL, VHDL) are very low-level
- Interfaces are often specific to vendors, platforms, FPGA boards
- Implementation (compile-equivalent) takes **hours**

Programmability improvements

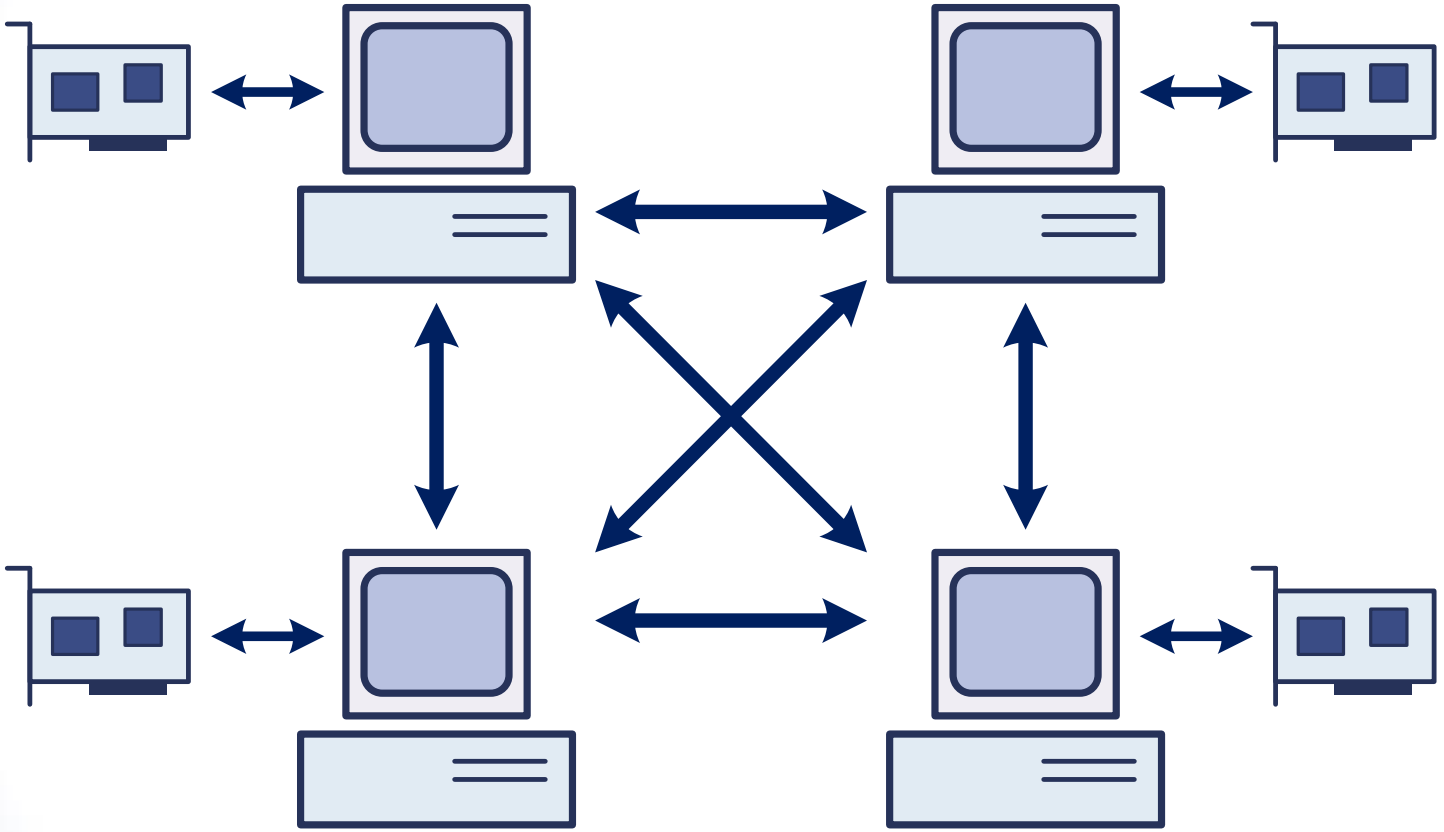
- Higher-level HDLs
  - SystemC, SystemVerilog, Bluespec, *Chisel (UC Berkeley)*
- High-Level-Synthesis ("C-to-gates")
  - Very active area in research and industry, but:
  - Only useful if programmer understands hardware

Programmability improvements (2)

OpenCL as an interface to FPGAs

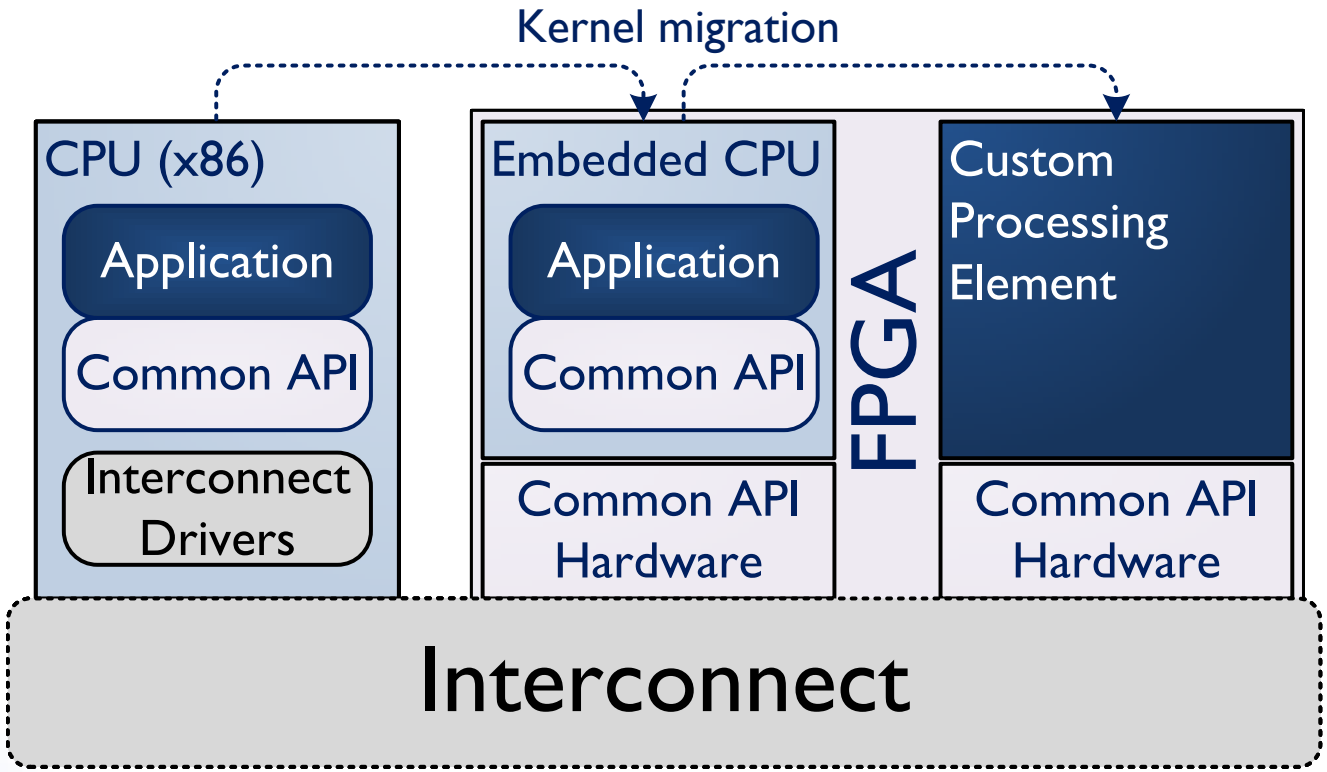
- Abstracts hardware interface as it does for GPUs
- Sets a programming model (memory, core hierarchy)
- Implies High-Level-Synthesis for actual code kernels
- But GPU model not a perfect fit for FPGA
  - Overconstrained dataflow

Classic accelerator model: Master-Slave



Our programming model philosophy

- Use a common API for Software and Hardware



Common SW/HW API

- CPU and FPGA components can initiate data transfers
- SW and HW components use similar call formats
- For distributed memory and message-passing, this was implemented by TMD-MPI
  (TMD: Toronto Molecular Dynamics)

- Our contribution:
  Building a heterogeneous infrastructure for PGAS

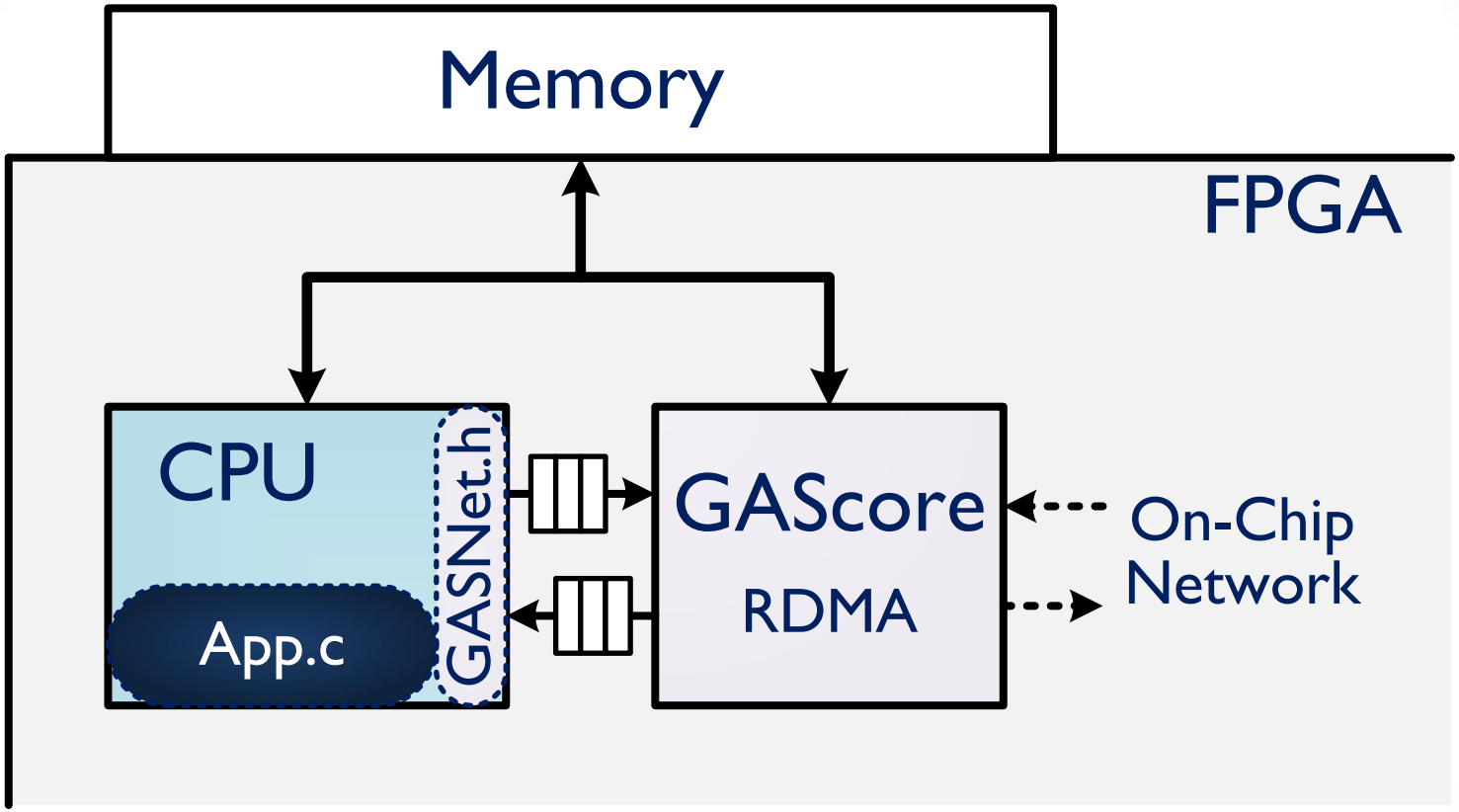
Why again a common API?

- Easier development: SW Prototyping → Migration
- Model makes no distinction between CPUs and FPGAs (in terms of data communication, synchronization)
- FPGA-initiated communication relieves CPU (even more so for 1-sided comm.)
- FPGA-only systems (or 1 CPU + many FPGAs) can work efficiently

THeGASNet

- Toronto Heterogeneous GASNet
- Compatible software and hardware implementations of the GASNet Core API
- GASNet is widely used, well-defined PGAS API
- Core API's "Active Message" types (S/M/L/LA) and handlers cover essential hardware needs

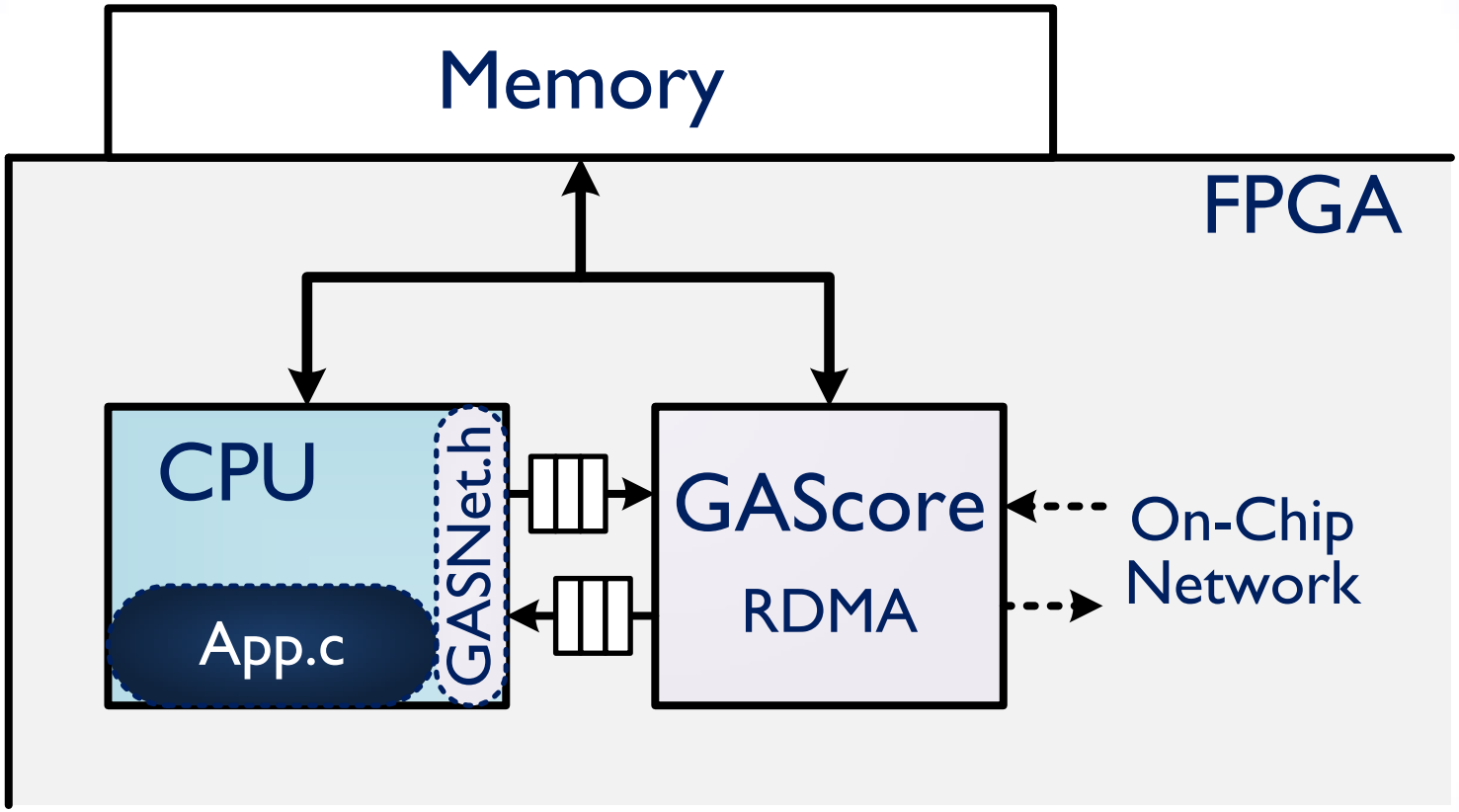
THeGASNet Hardware: GAScore



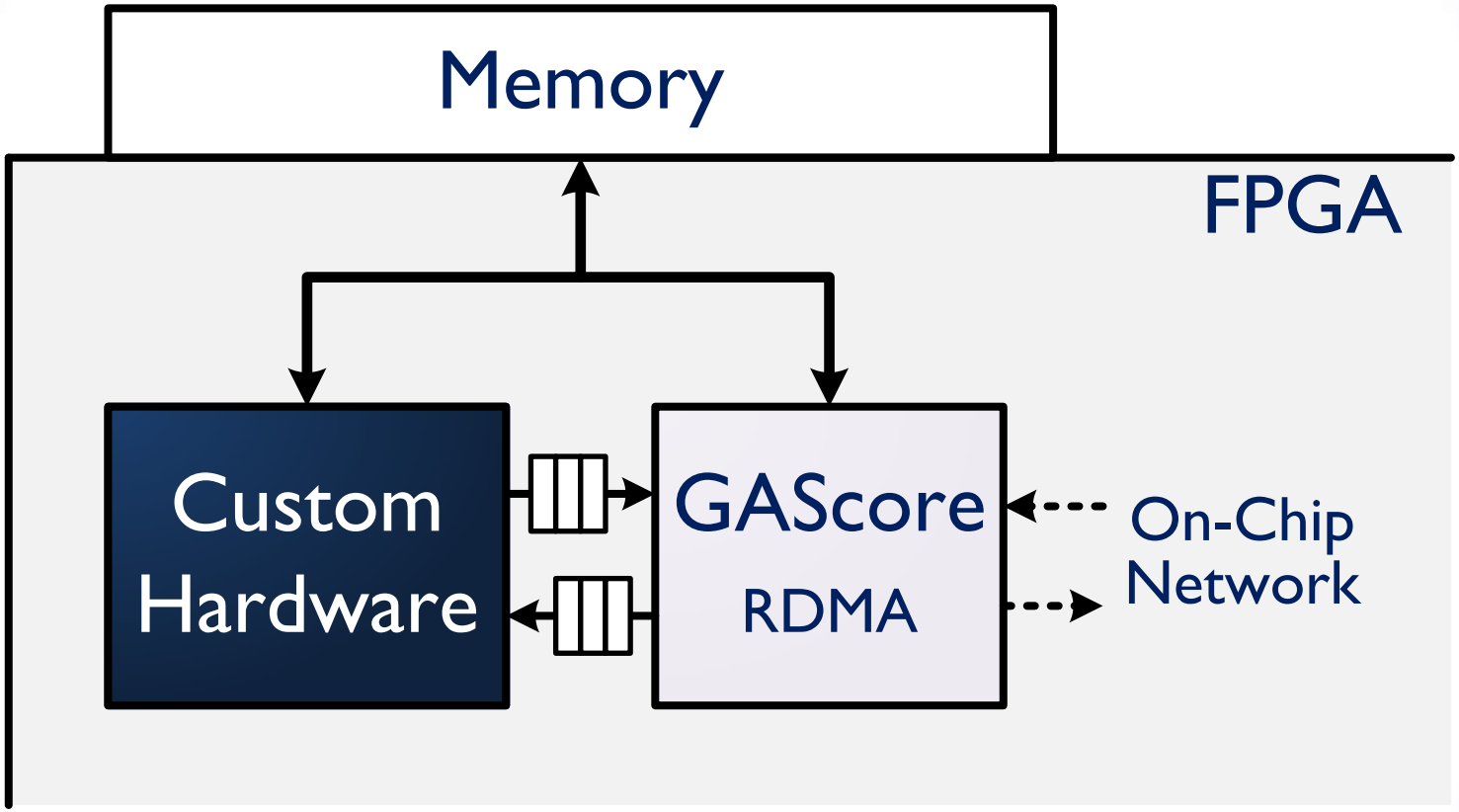
GAScore

- Remote memory communication engine (RDMA)
- Controlled through FIFOs (FSLs)
- Configuration parameters are the same as for GASNet Active Message function calls
  - Simple software layer for embedded CPUs

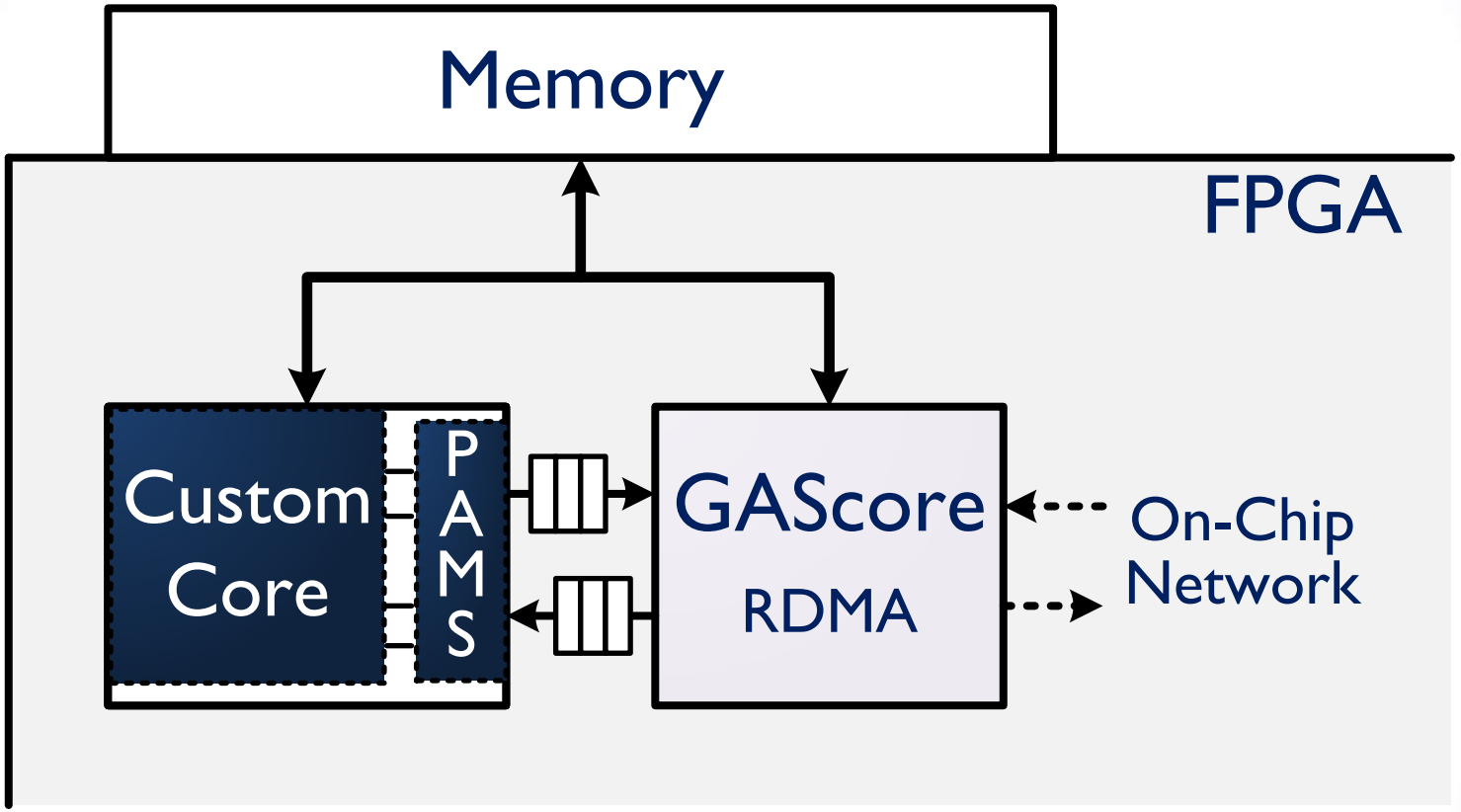
THeGASNet Hardware: GAScore



THeGASNet Hardware: GAScore



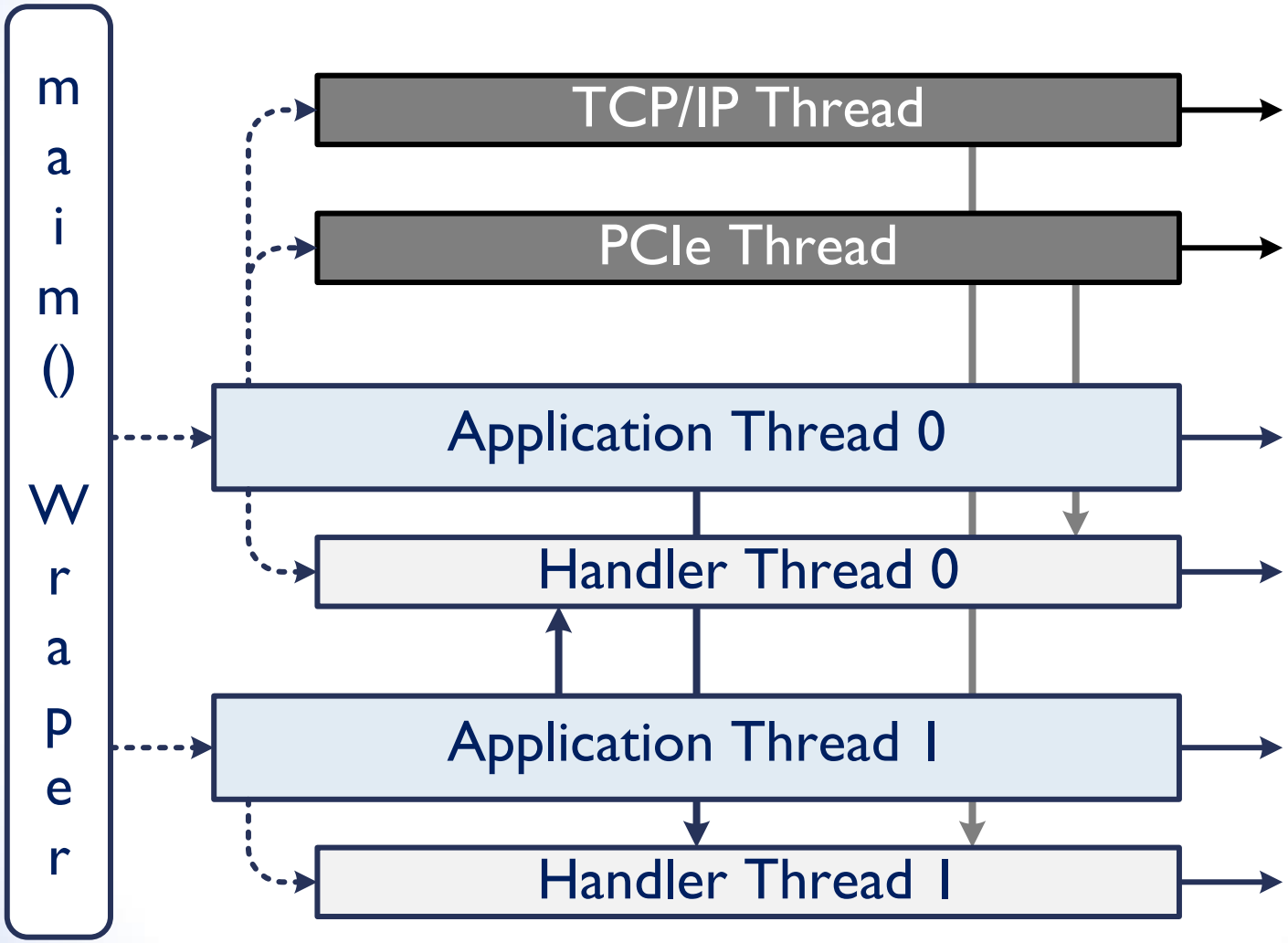
THeGASNet Hardware: PAMS



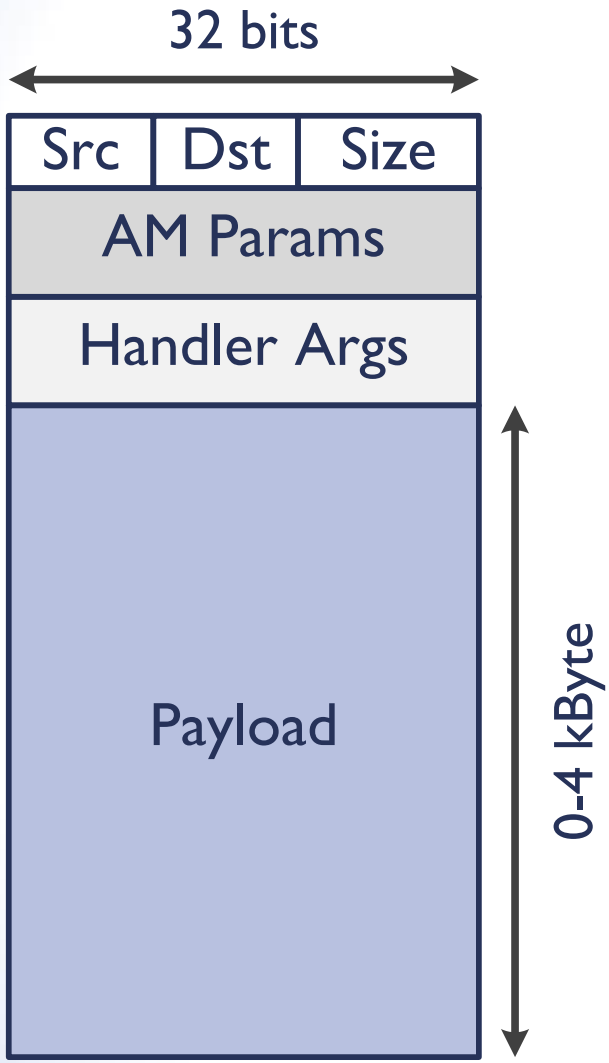
Programmable Active Message Sequencer

- Controls custom hardware operations
- Handles reception/transmission of Active Messages
- Custom hardware ops and Active Messages are initiated based on:
  - Custom hardware state
  - Number of received messages with a specific code
  - Amount of received data
- (Re-)programmable through Active Messages
- Future: Reconfigure Computation Core by AM

THeGASNet Software (PC/ARM)



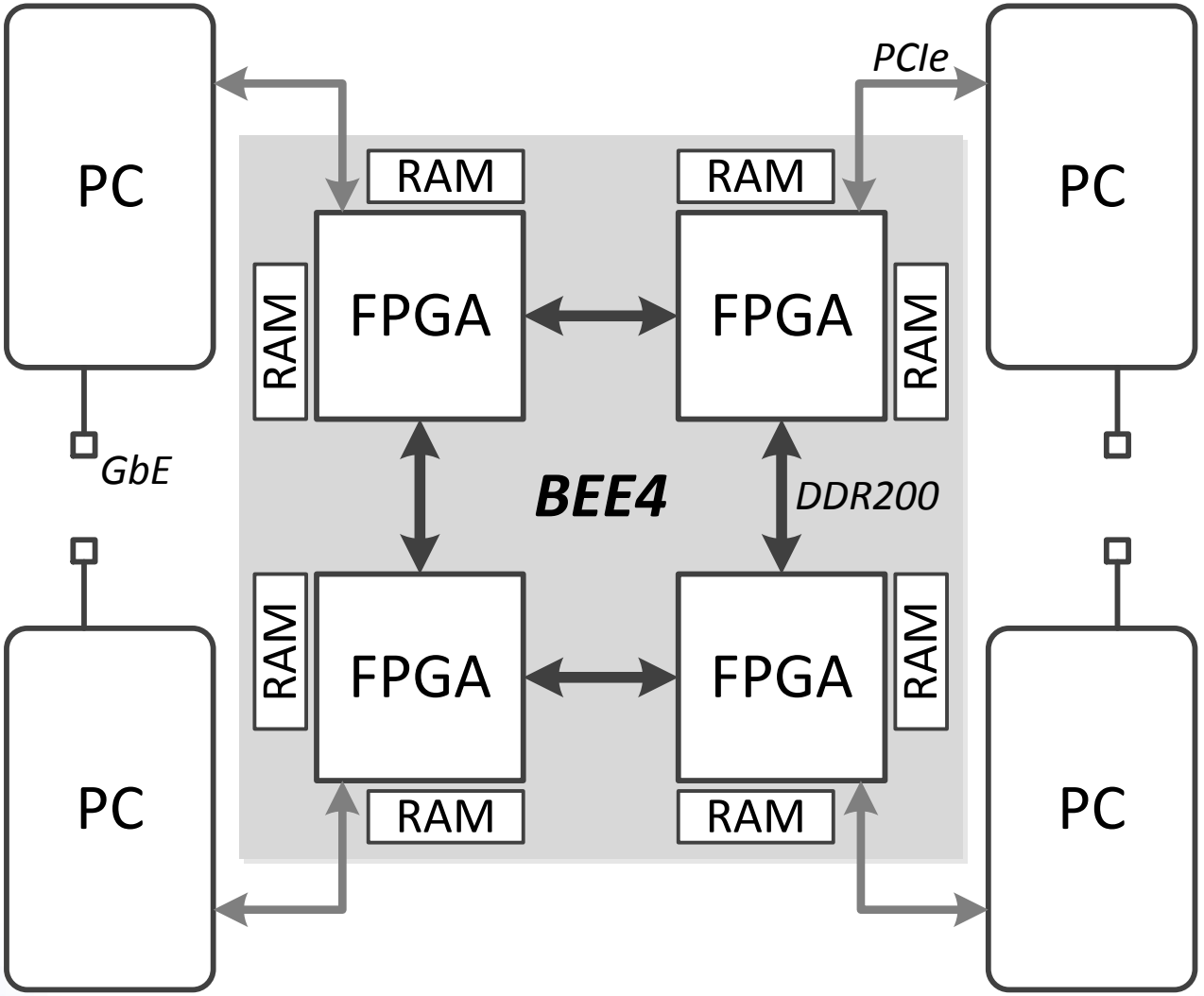
THeGASNet packet format



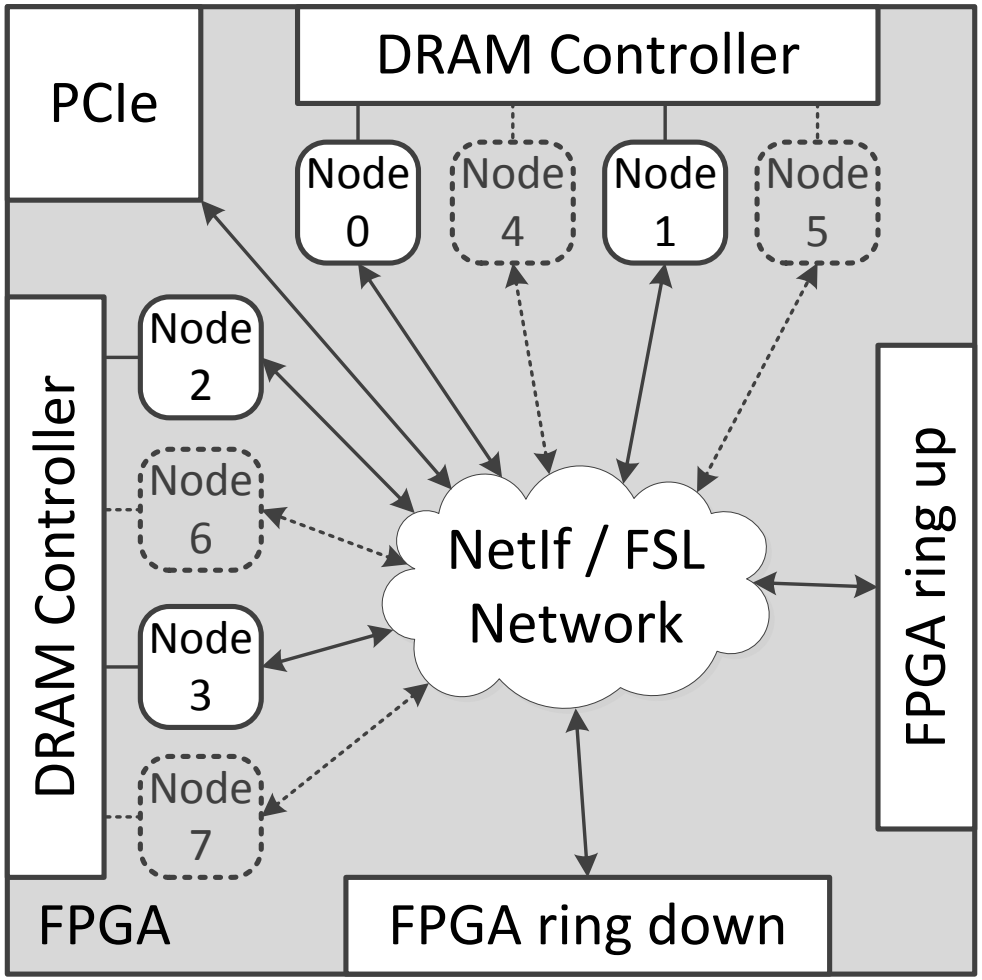
Identical for:

- Thread-to-thread
- TCP/IP
- PCIe
- On-chip network

MapleHoney - Heterogeneous Cluster



MapleHoney - Single FPGA



Microbenchmarks: FPGA

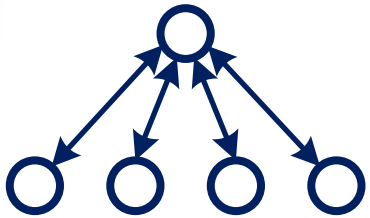
Short message latency (ns)

FPGA hops	1-way	2-way
0	160	330
1	240	490
2	320	550

Long message latency (ns)

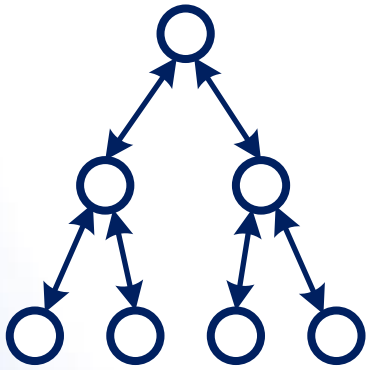
FPGA hops	4 Bytes	4 kBytes
0	540	22555
1	620	22635
2	700	22715

Microbenchmarks: FPGA



Simple barrier latency (ns)

Latency to node 0	870
Latency to node 0 and back	1960



"Staggered" barrier latency (ns)

Latency to node 0	620
Latency to node 0 and back	1480

Microbenchmarks: PC,TCP/IP,1GEth

Full barrier latency (4x 4 Threads)	21.6 ms
2-way Short message latency	9.3 ms
1-way 4 kByte Long message latency	6.2 ms
Average Bandwidth 64MByte transfer	0.84 Gbit/s
Average Bandwidth 1GByte transfer	0.87 Gbit/s

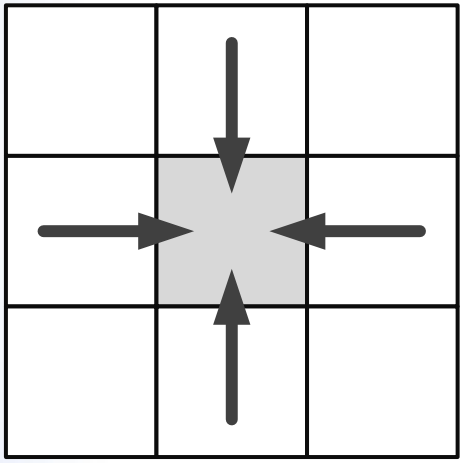
! High variation on latency measurements except barrier => Thread scheduling issues?

Microbenchmarks: PC <-> FPGA (PCIe)

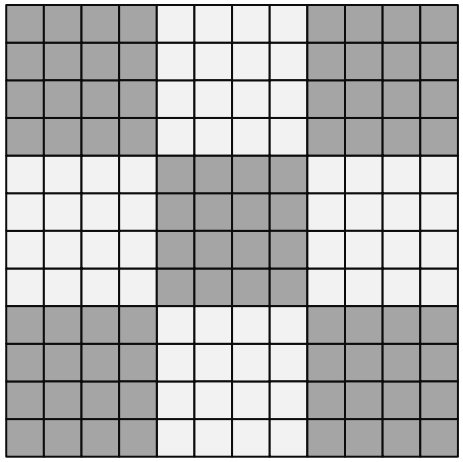
2-way Short message latency	15.7 us
1-way 4 kB Long latency PC -> FPGA	32.4 us
1-way 4 kB Long latency FPGA -> PC	23.6 us
Sustained PCIe bandwidth PC -> FPGA	175.1 Mbyte/s
Sustained PCIe bandwidth FPGA -> PC	172.3 Mbyte/s

Application example: Jacobi Heat Transfer

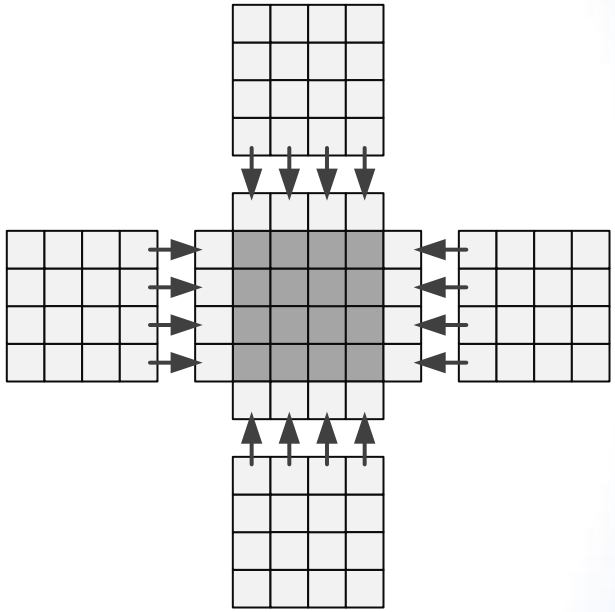
Iterative PDE solver with stencil computation pattern



Cell iteration



Node division



Communication pattern

Step I: Design PC application

- Design SPMD application using THeGASNet
- Problem size: 45760x45760 cells on 16 nodes/threads
- 11880x11880 cells per node/thread: ~1GB storage
- Debug and profile application on PC platform

Step II: Migrate to MicroBlaze

- Using the same THeGASNet headers and compiler family (gcc), differences should be minimal
- Static instead of dynamic memory allocation; shared segment pre-defined.
- OS functionality like timers may need to be changed (abstracted in platform header)
- Command-line arguments to application converted to compile-time arguments (application unchanged)

Step III: Replace MicroBlaze with HW

- External connections (Memory bus and GAScore interface) stay the same
- Custom core only does computation
- PAMS needs to be configured for communication in-between computation phases
- Keeping one SW GASNet thread on PC to
  - Move data into FPGA memory
  - Program PAMS
  - Output benchmark time

Performance comparison

Time for one complete iteration (averaged)

4 PCs, 16 threads	4.54s
16 MicroBlazes	222.86s
16 Custom FPGA cores	5.69s

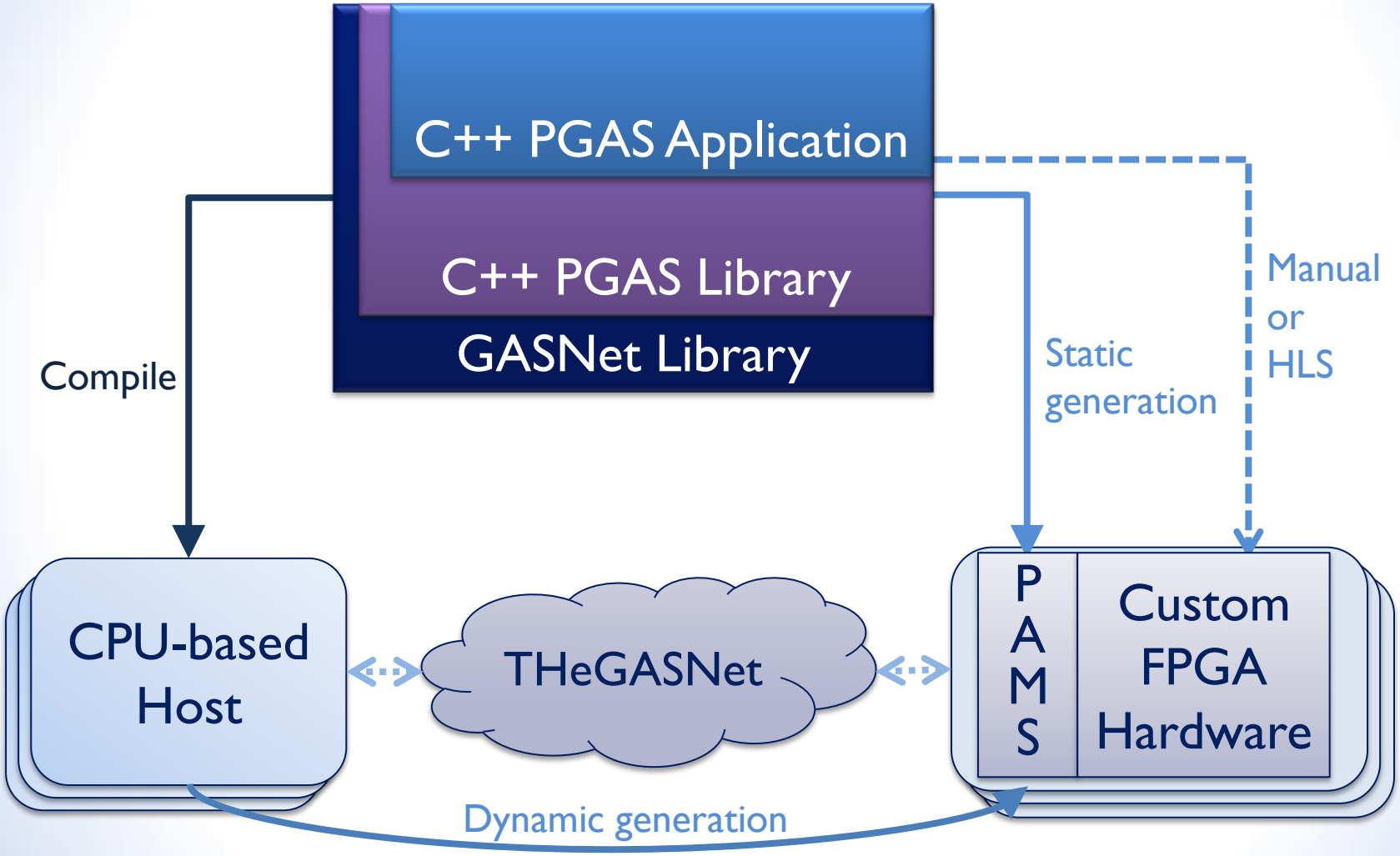
Memory-bound: No FPGA gain!

Conclusion

GASNet SW API + GAScore (+PAMS) enable

- Universal abstraction for FPGA interfaces
- Simple heterogeneous communication and synchronization
- Simplified kernel migration to accelerators
- Still have to pick the right problems for FPGA speedup!

Outlook: THePaC++
Heterogeneous C++ PGAS library



Thank you for your attention!

Questions?