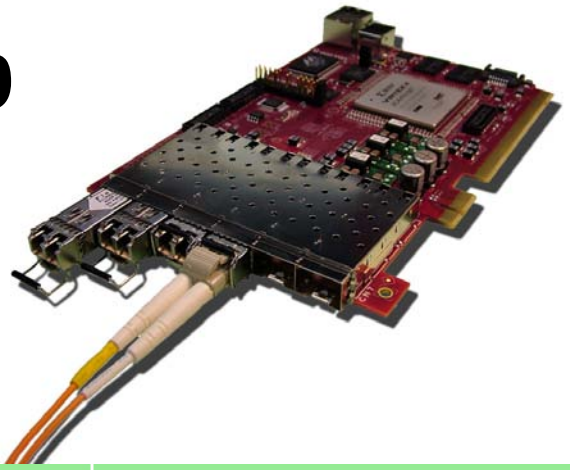
RUPRECHT-KARLS-
UNIVERSITÄT
HEIDELBERG

Leveraging HyperTransport for a custom high-performance cluster network

Mondrian Nüssle

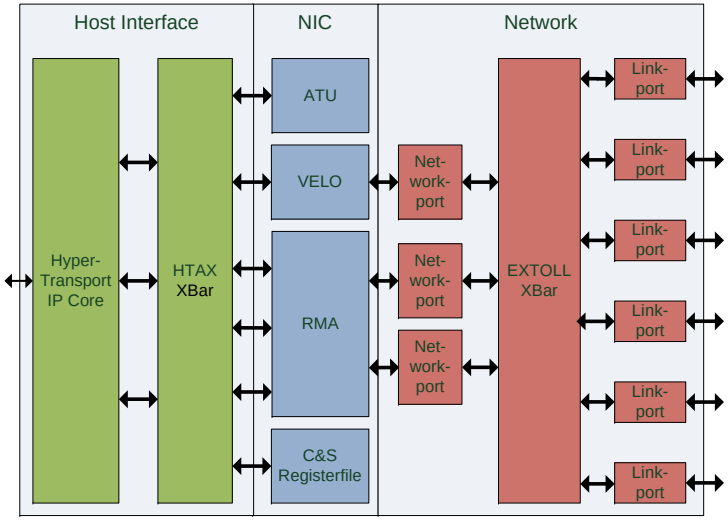
HTCE Symposium 2009

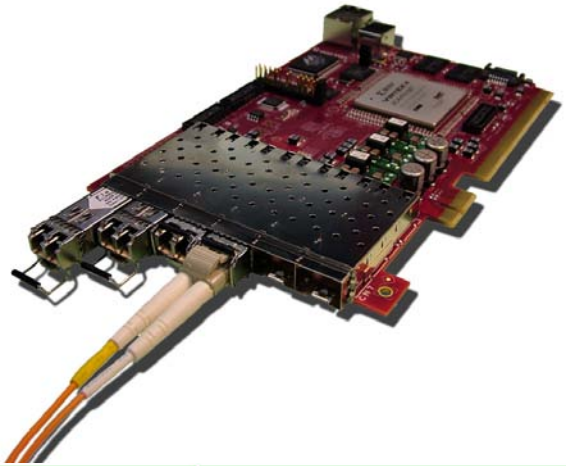
11.02.2009



Outline

- Background & Motivation
- Architecture
- Hardware Implementation
- Software Stack
- Results
- Conclusion



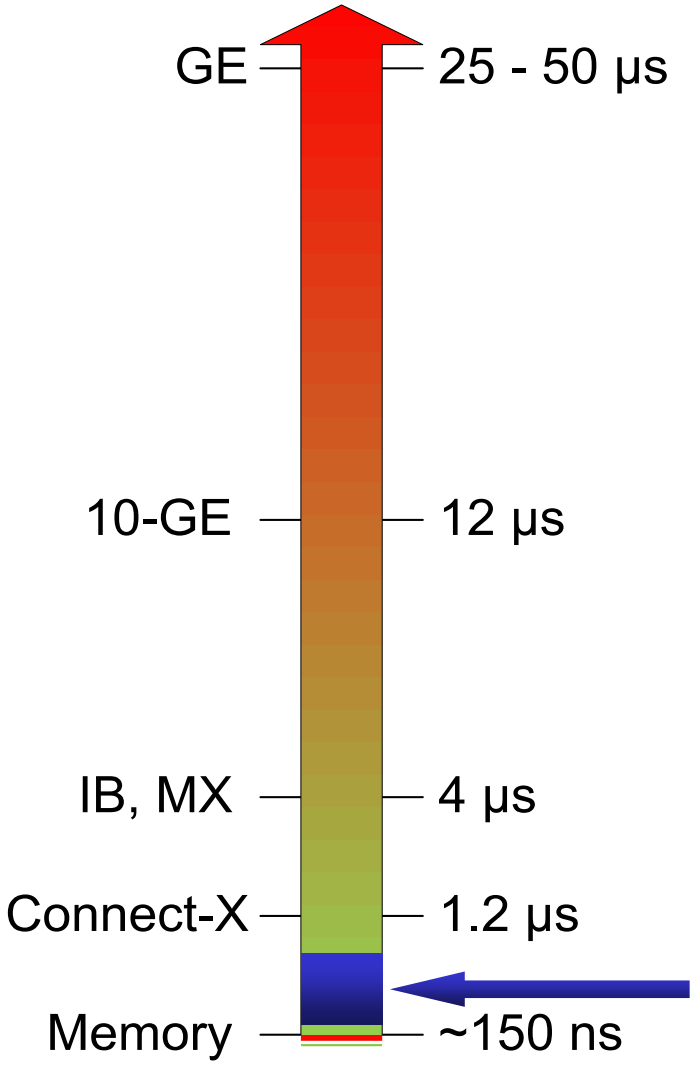


EXTOLL: Background & Motivation

- High-performance computing synonymous with parallel computing
- Interconnection networks between processors are a **key** component in parallel systems
- Patterson stated: "*Latency lags Bandwidth*"

- The **EXTOLL** project at the CAG aims to significantly **lower communication latency** and improve communication in parallel systems

Goals



- Enable communication with extremely low latency
  → close to main memory access
- Enable communication – computation overlap
- Design a balanced system
  - In terms of CPU on-loading and off-loading
  - In terms of system complexity

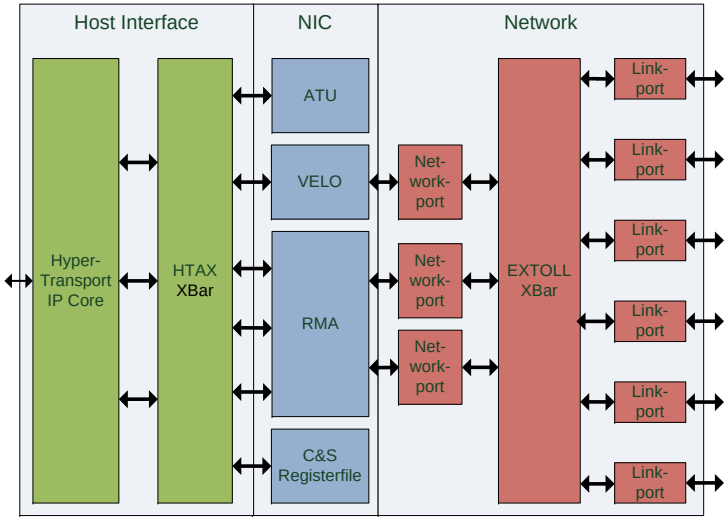
➢ Adding bandwidth is much easier ☺

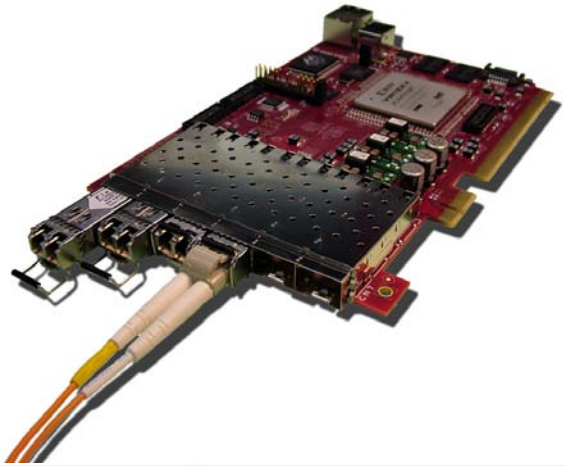
Key design facts

- Leverage HT as host interface for lowest latency of data transport between CPU and device
- Leverage modified HT as on-chip communication protocol
- Implement a lean network interface controller:
  - Minimize state information on NIC
  - Provide user-level, virtualized access (avoid kernel)
  - Minimize number of CPU ↔ device and memory ↔ device transactions
- Network layer that provides reliable, in-order, low-latency transport service

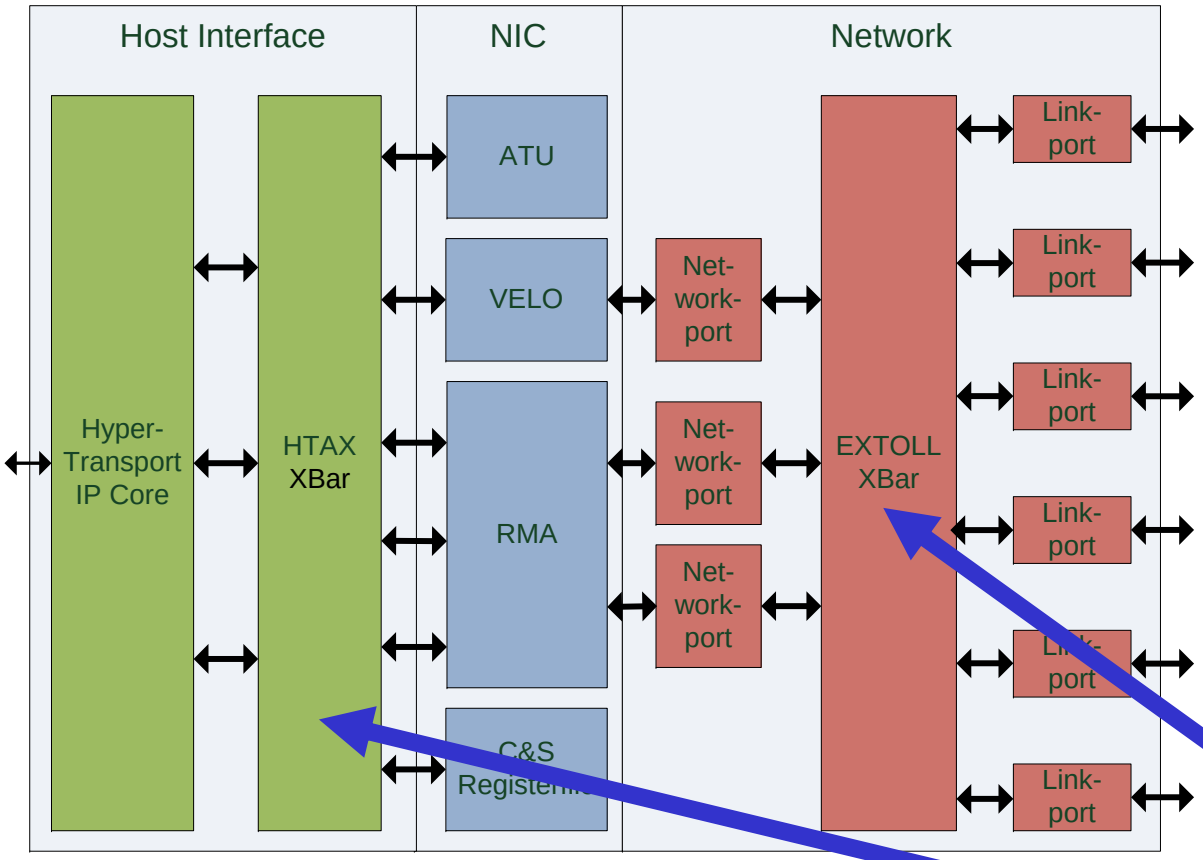
Outline

- ✓ **Background & Motivation**
- ❑ Architecture
- ❑ Hardware Implementation
- ❑ Software Stack
- ❑ Results
- ❑ Conclusion





Block diagram

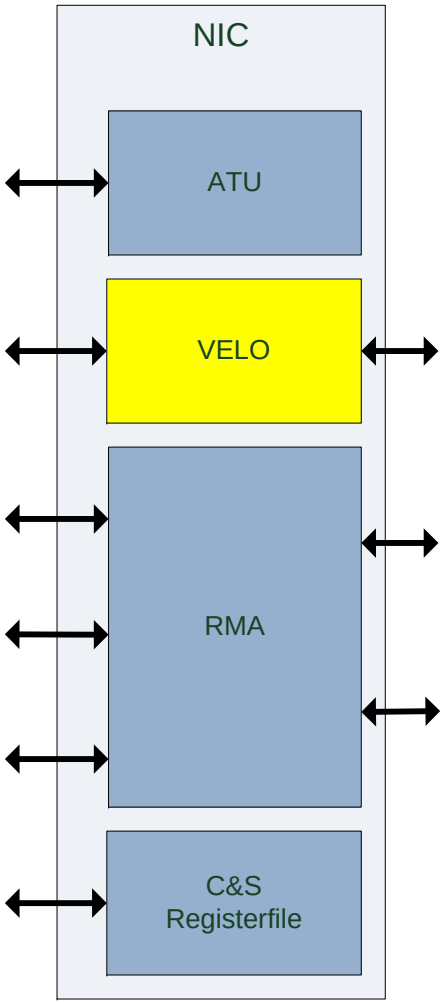


- Host Interface block
- NIC block:
  - Several communication functions
- Network block
  - 6 links
  - 9x9 crossbar

Flexible architecture:

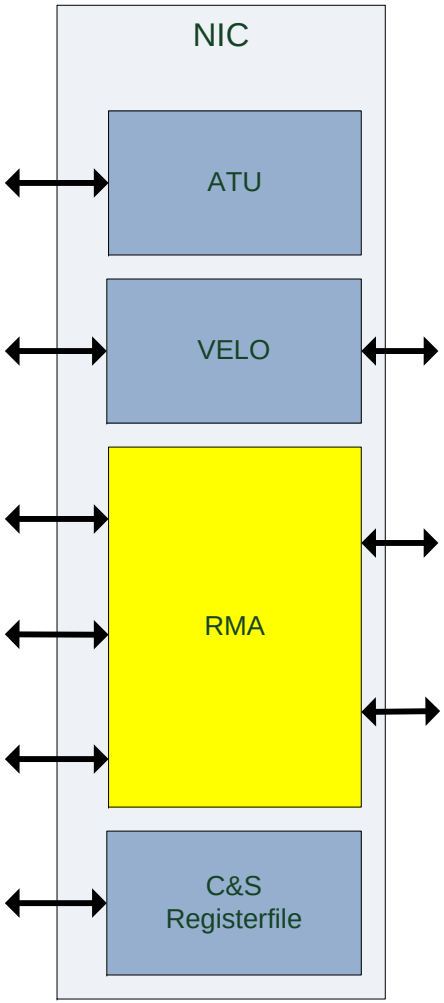
Configurable data path

Communication functions: VELO



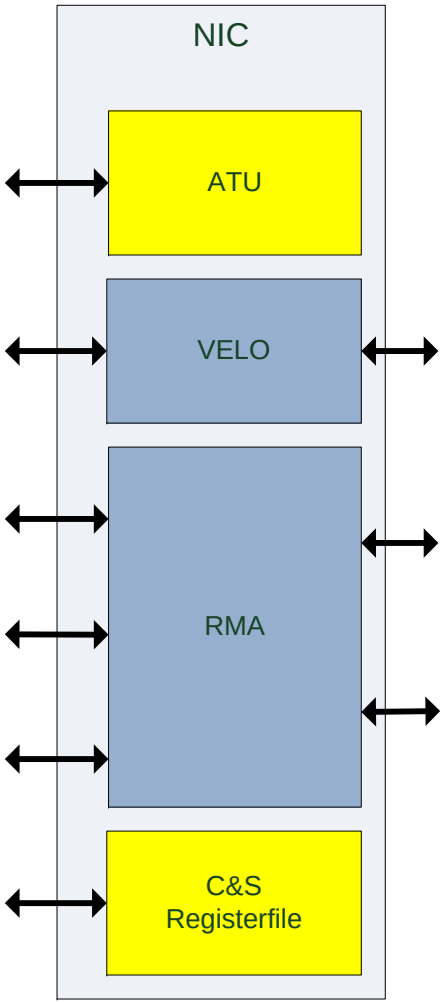
- Virtualized Engine for low overhead
  - Enable ultra-low send/receive communication
  - Supports messages of up to 64-byte (one cache line) directly
  - A single PIO transaction triggers sending of a message
  - Message completion at the receiver is usually performed with a single DMA transaction
- Minimized traffic between host and device!

Communication functions: RMA



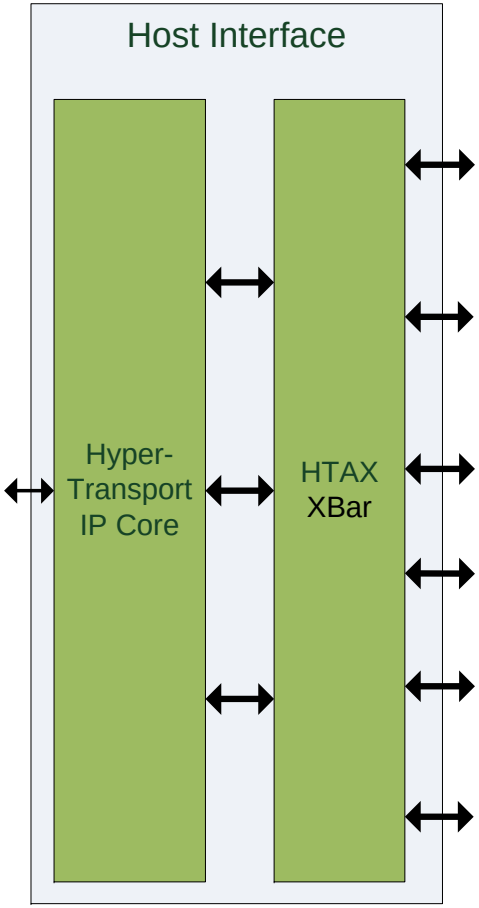
- EXTOLL Remote Memory Architecture
  - Enables access to remote memory using put, get and atomic transactions
  - Transaction triggered by a single 128-bit SSE2 store
    → minimizing start-up latency
  - Flexible notifications:
    - at the requester
    - the completer
    - the responder
    - or any combination thereof

Supporting modules



- Address Translation Unit
  - Provides address translation services for RMA
  - Registration/unregistration latency in prototype systems starts at **~2 µs**
  - Translation using on-chip TLB and main-memory tables
- Control and Status Registerfile
  - automatically generated from high-level spec (including kernel code)
  - Local and remote access possible (network management software)

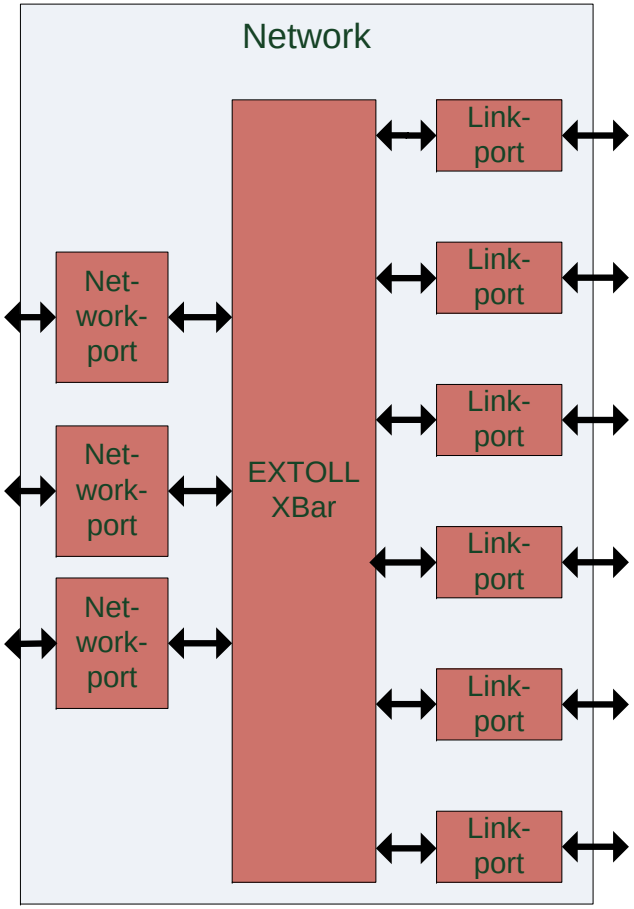
HT Interface



- HT-Core: interface to host
- All functional units need to communicate with host
- Avoid protocol conversion for on chip-network

→ HTAX crossbar running on-chip protocol
  - simplified
  - more source tags
  - fixed format

Network layer



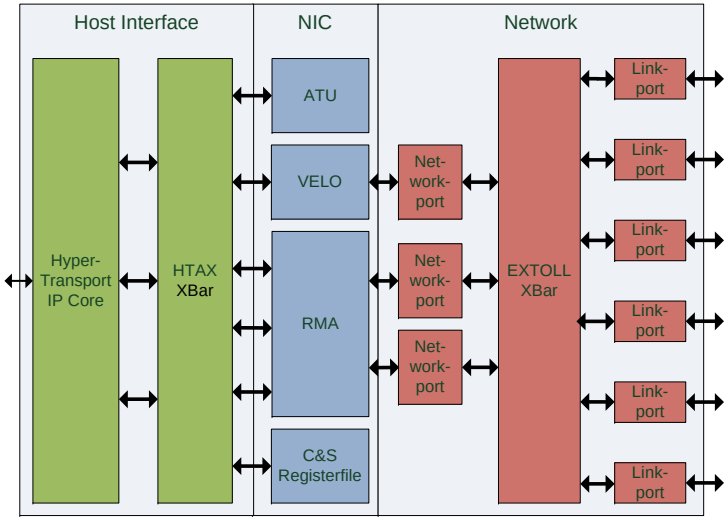
- Fully parametrizable width of data-paths and number of ports
- In-order delivery of packets
- Virtual channels
- Hardware retransmission
- Cut-through switching
- Credit based flow-control

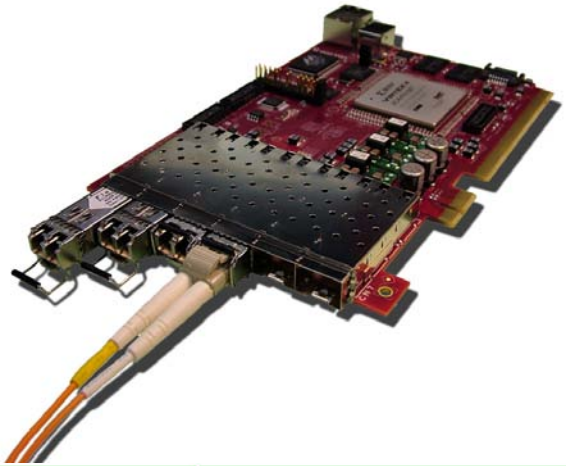
Current implementations:

- 6 ports used to connect to external links
- 16+2 bit data path width

Outline

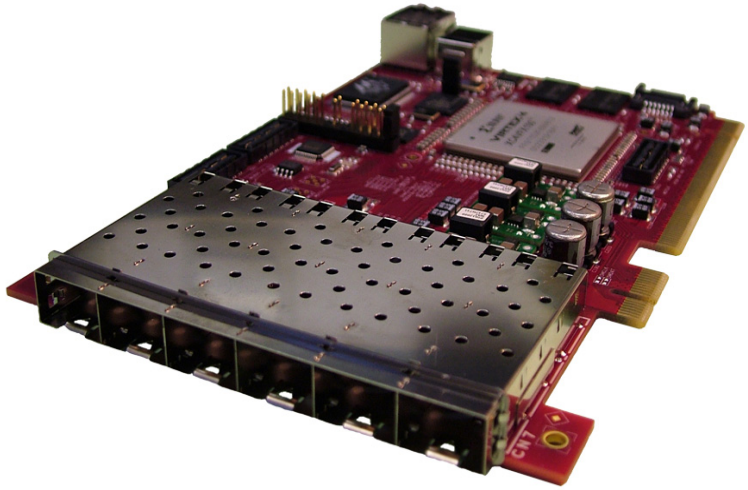
- ✓ **Background & Motivation**
- ✓ **Architecture**
- ❑ Hardware Implementation
- ❑ Software Stack
- ❑ Results
- ❑ Conclusion





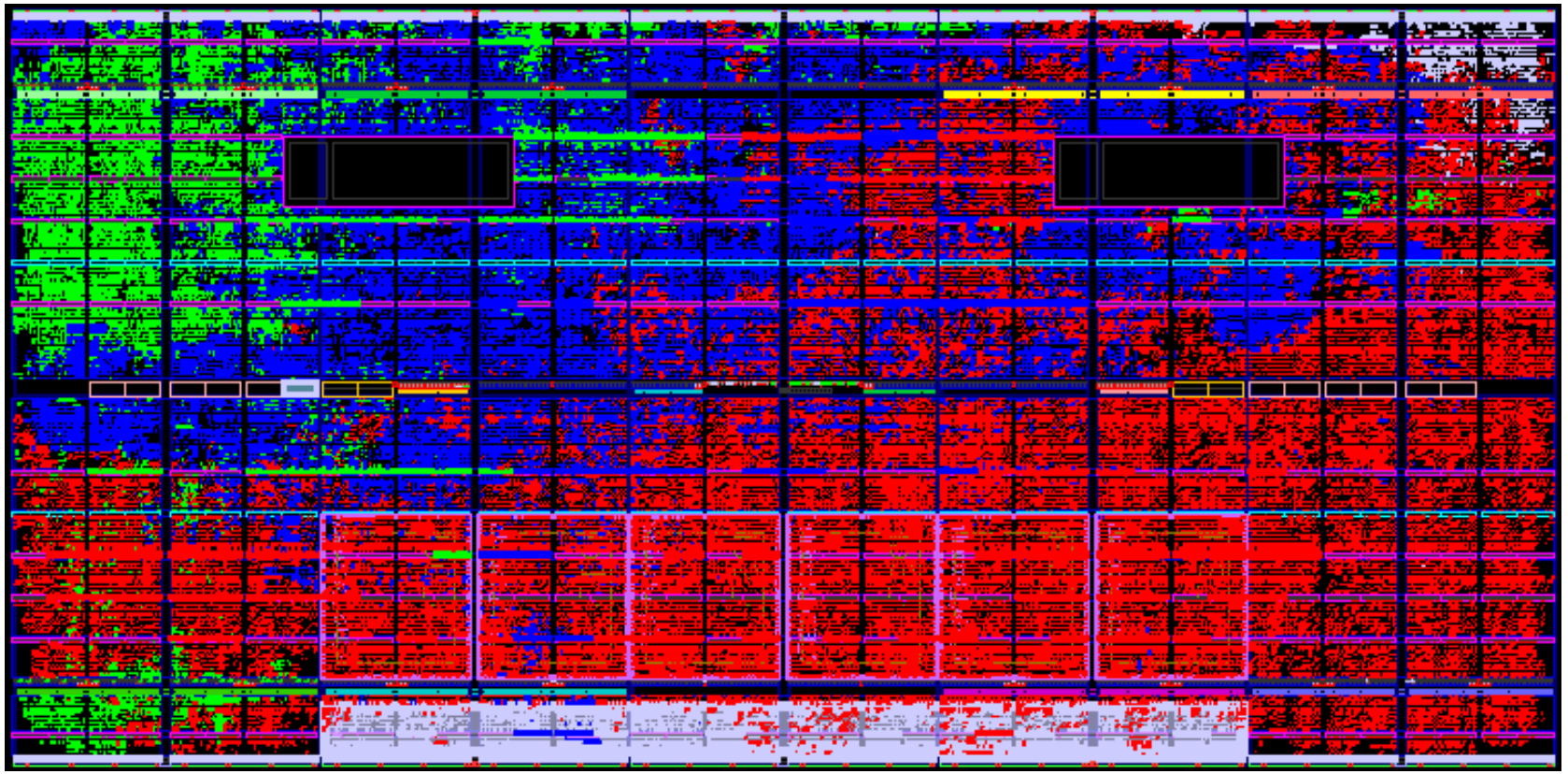
Implementation I

- EXTOLL prototype is implemented on the HTX-Board
  - Virtex 4 FX100 FPGA, speed-grade 11 or 12
  - 6 SFP optical transceivers



- Currently :
  - 16 bit width, 180 MHz core frequency
    - 3.6 Gb/s links

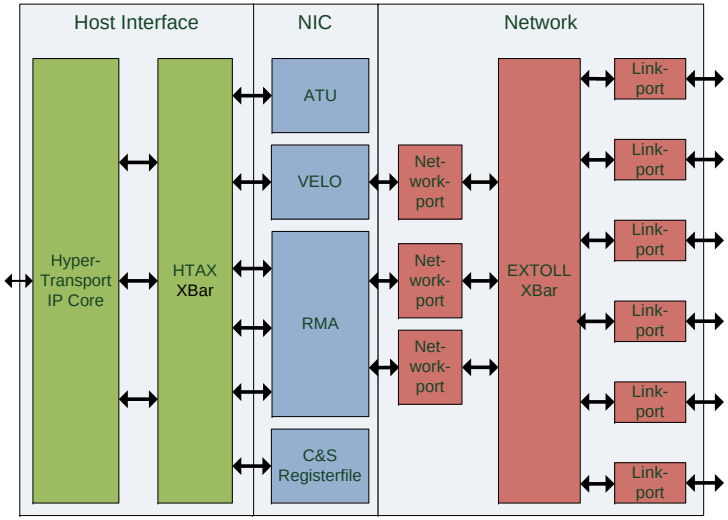
Implementation II

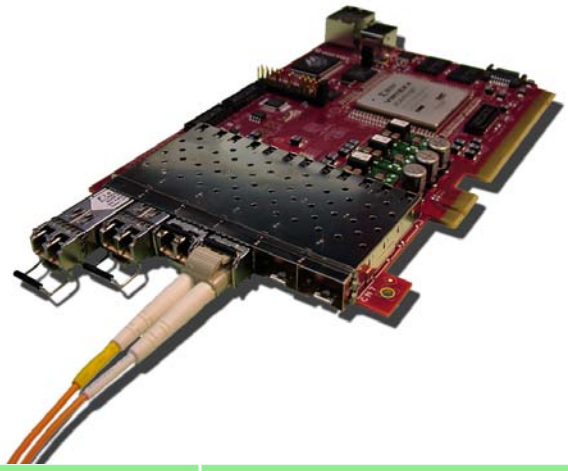


- > 90% of all slices of the FPGA are in use for the design
- HT-Core runs at 200 MHz internal frequency and HT400
- EXTOLL modules run with 180 MHz on speed-grade -12 device

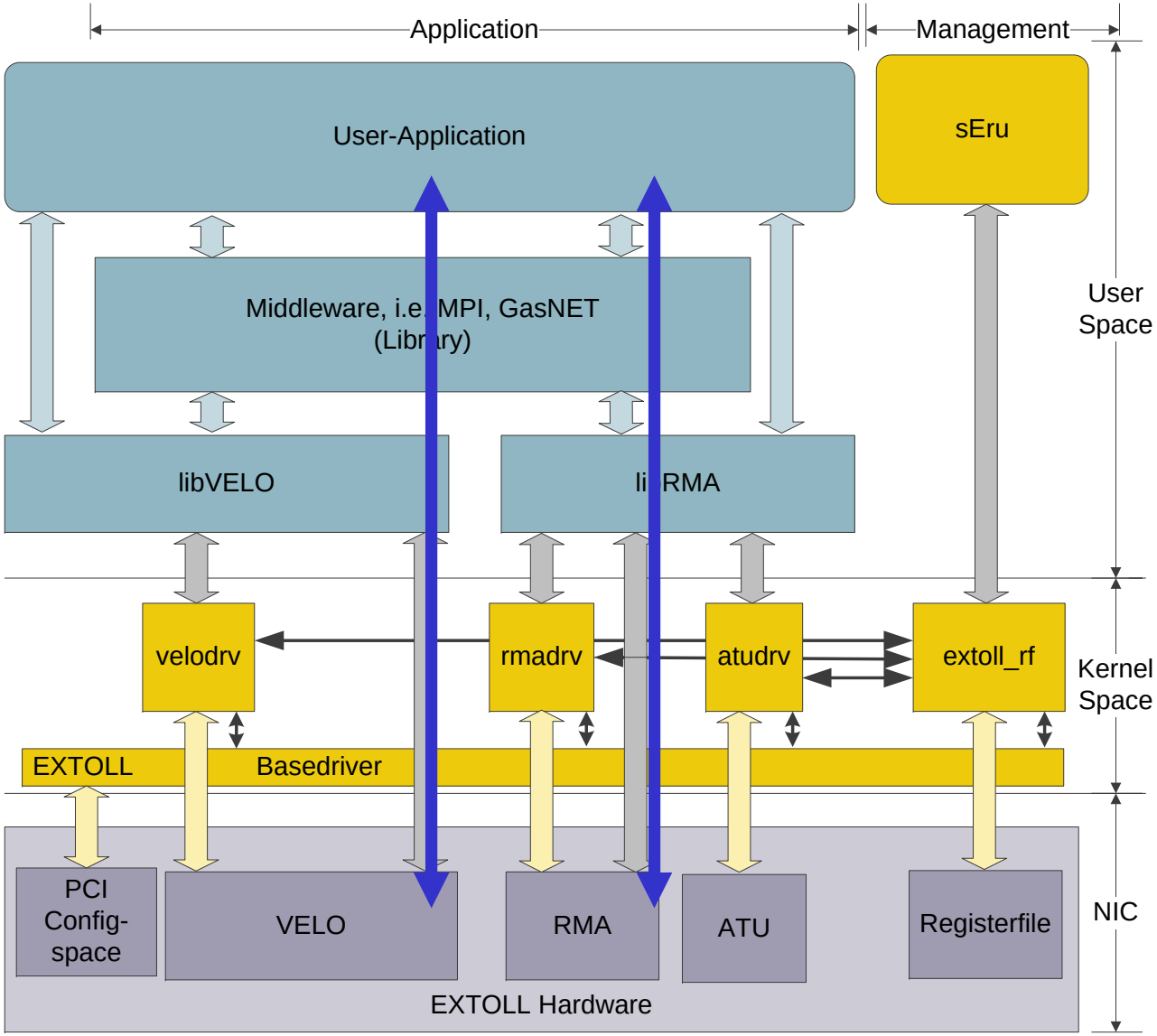
Outline

- ✓ **Background & Motivation**
- ✓ **Architecture**
- ✓ **Hardware Implementation**
- ❑ Software Stack
- ❑ Results
- ❑ Conclusion





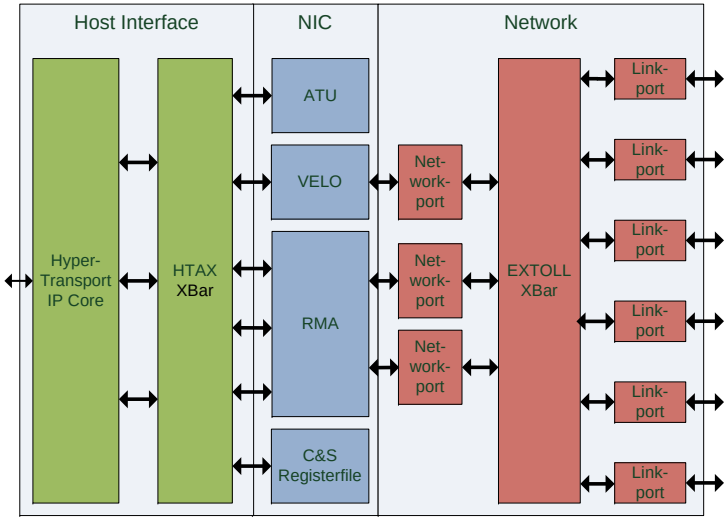
Software Stack

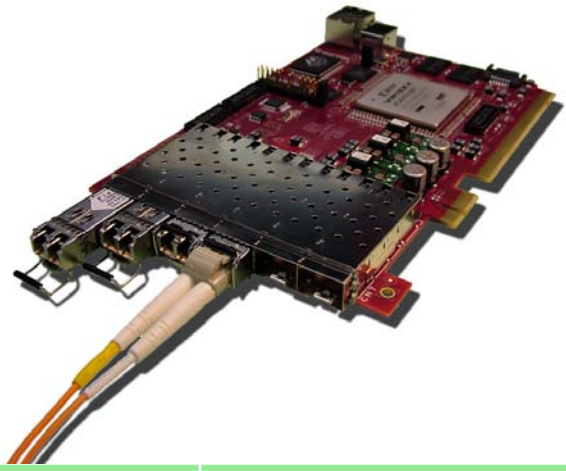


- OS bypass
- Layered approach
- PGAS support through GasNET
- MPI support through OpenMPI
- Linux kernel driver

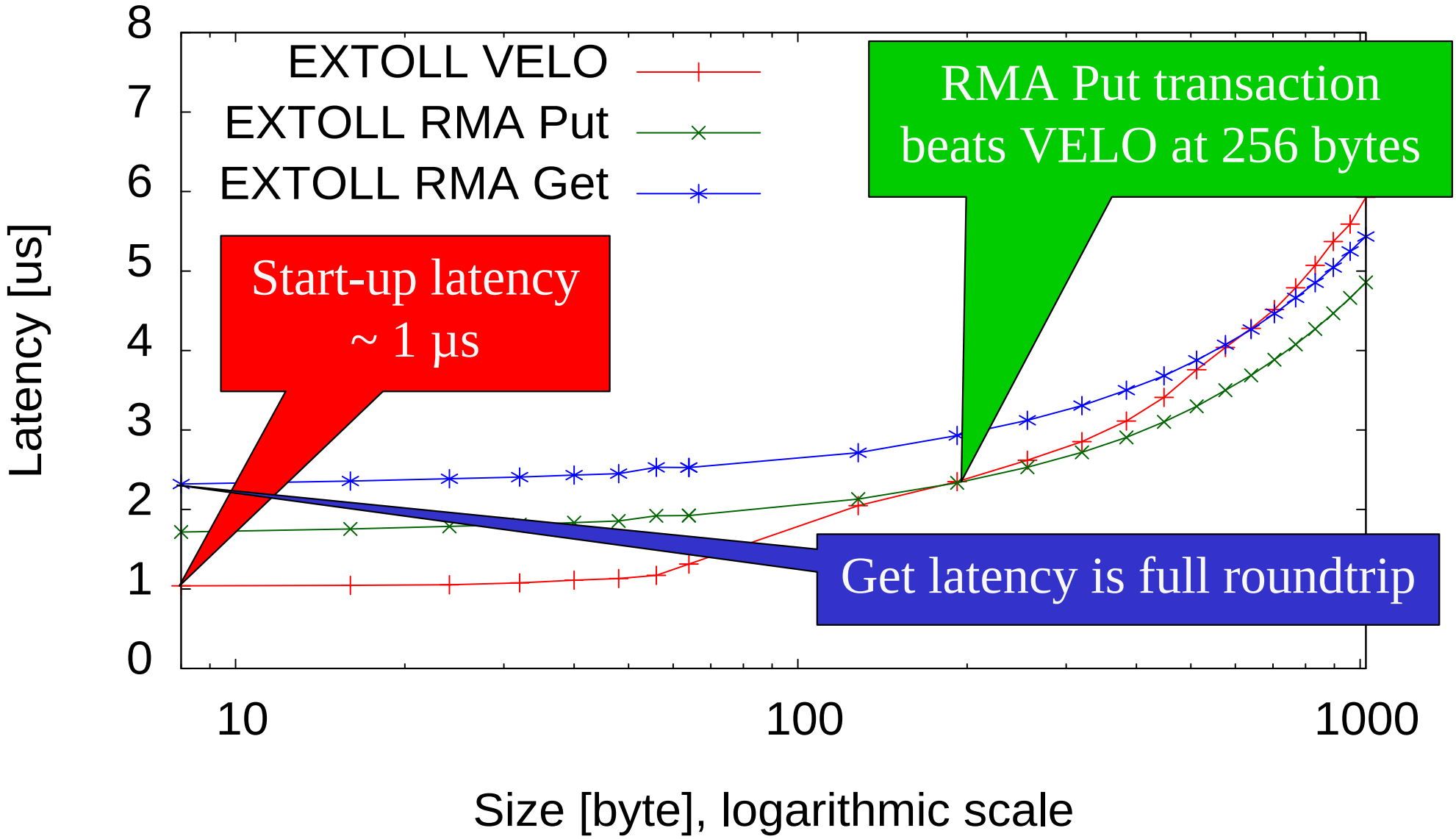
Outline

- ✓ **Background & Motivation**
- ✓ **Architecture**
- ✓ **Hardware Implementation**
- ✓ **Software Stack**
- ❑ Results
- ❑ Conclusion

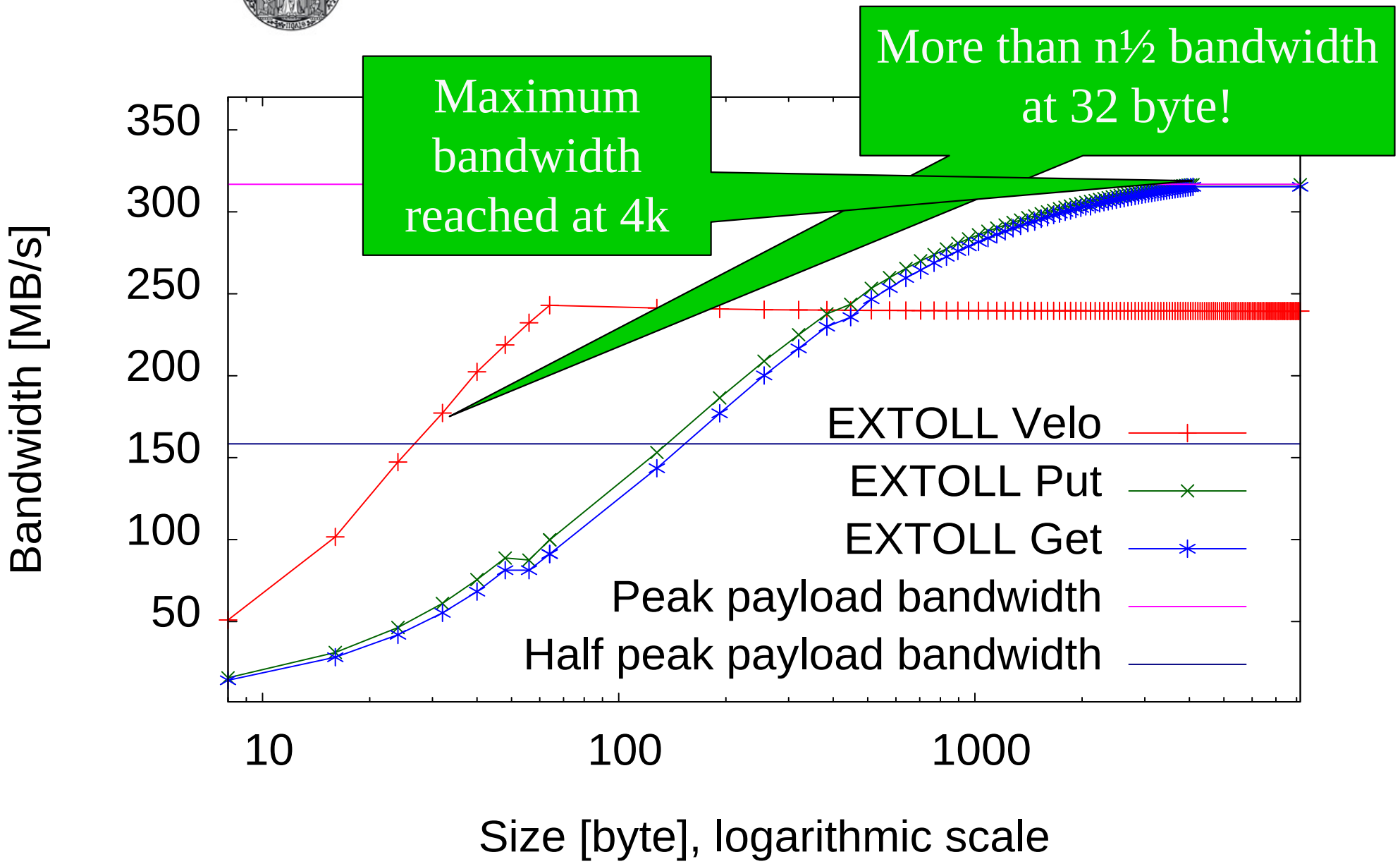




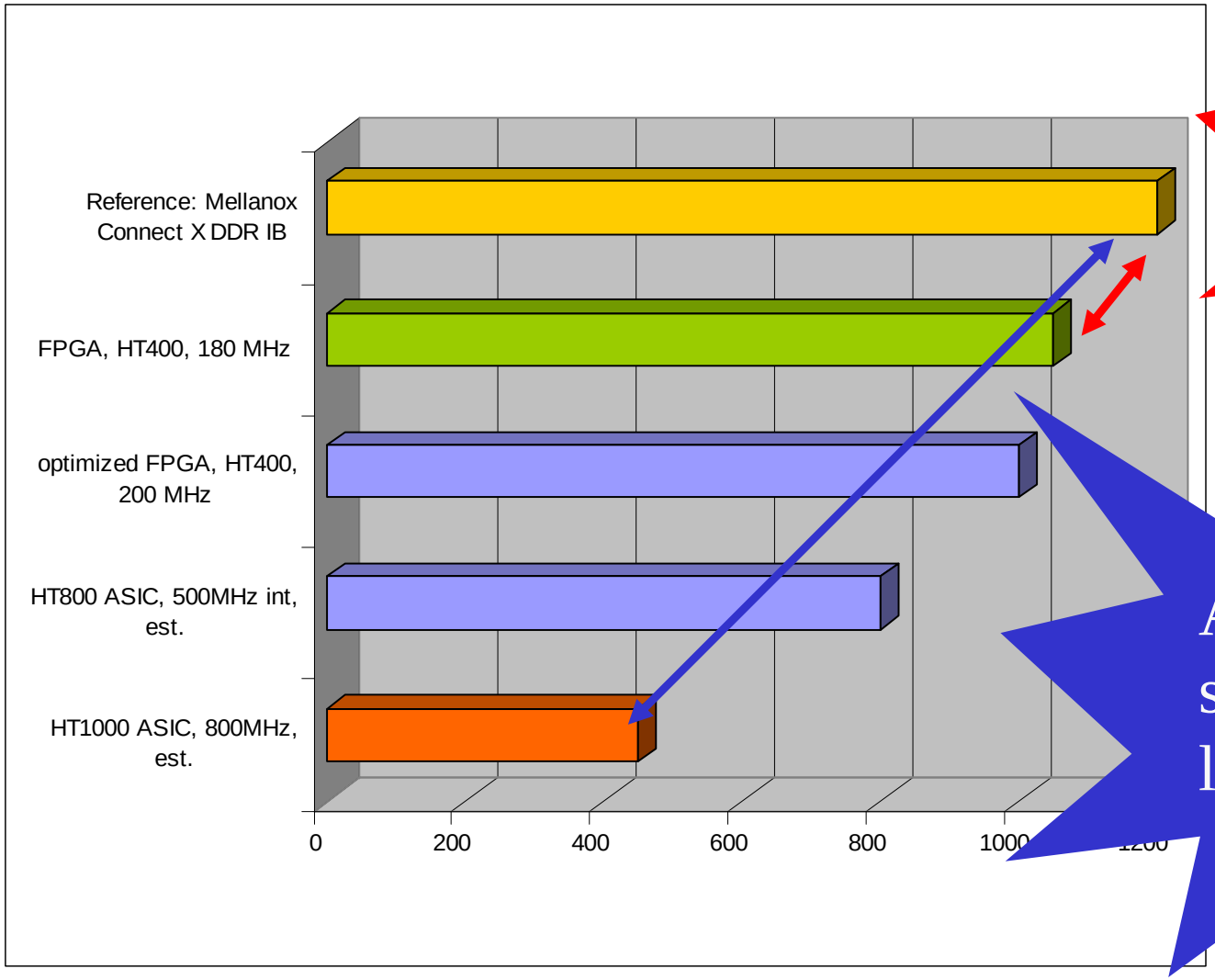
Results – Latency



Results - Bandwidth



Technology Scaling

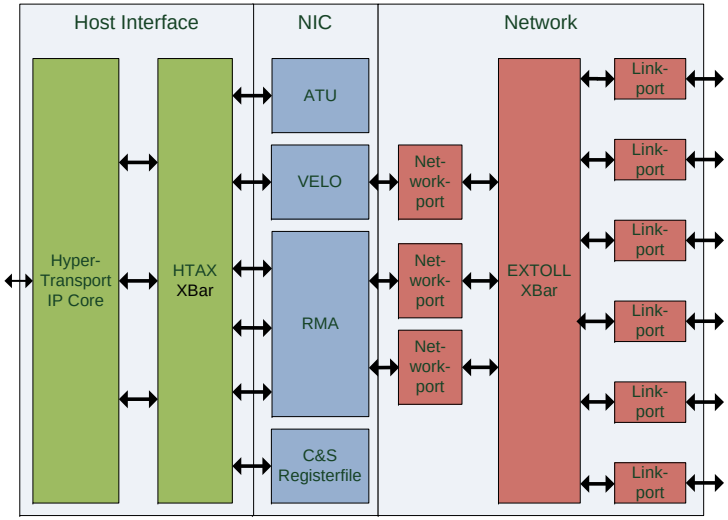


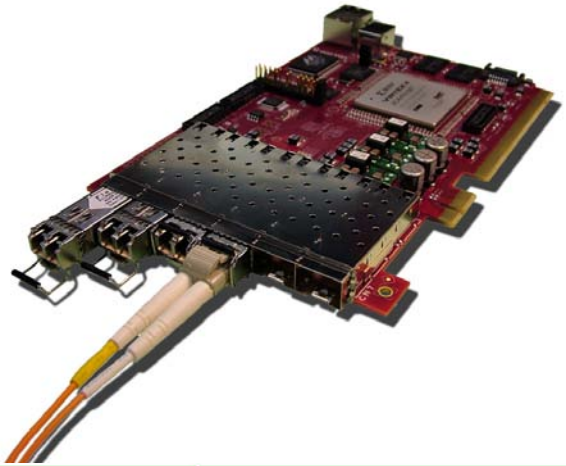
Already beats best IB Silicon

ASIC would show 3 times lower latency!

Outline

- ✓ **Background & Motivation**
- ✓ **Architecture**
- ✓ **Hardware Implementation**
- ✓ **Software Stack**
- ✓ **Results**
- ❑ Conclusion





Conclusion

- EXTOLL is an architecture for ultra low-latency communication in parallel systems
- prototype hardware is up and running
- basic software environment is up and running
- Performance numbers are excellent:
  - ~ **1 µs start-up latency** on FPGA prototype
  - Bandwidth limited by serializers & board, but can be improved with new platform

Next Steps

- more software is being added
  - Most interesting GasNET
- Evaluation on 1024-core Valencia Cluster
- On the hardware-side, next step is a new revision with a more powerful base technology

➢ Evaluation of next platform for HW

Thanks !

Questions?