RUPRECHT-KARLS-
UNIVERSITÄT
HEIDELBERG

Investigating Low Latency Trading with an HT enabled FPGA

Benjamin Geib
Computer Architecture Group
University of Heidelberg

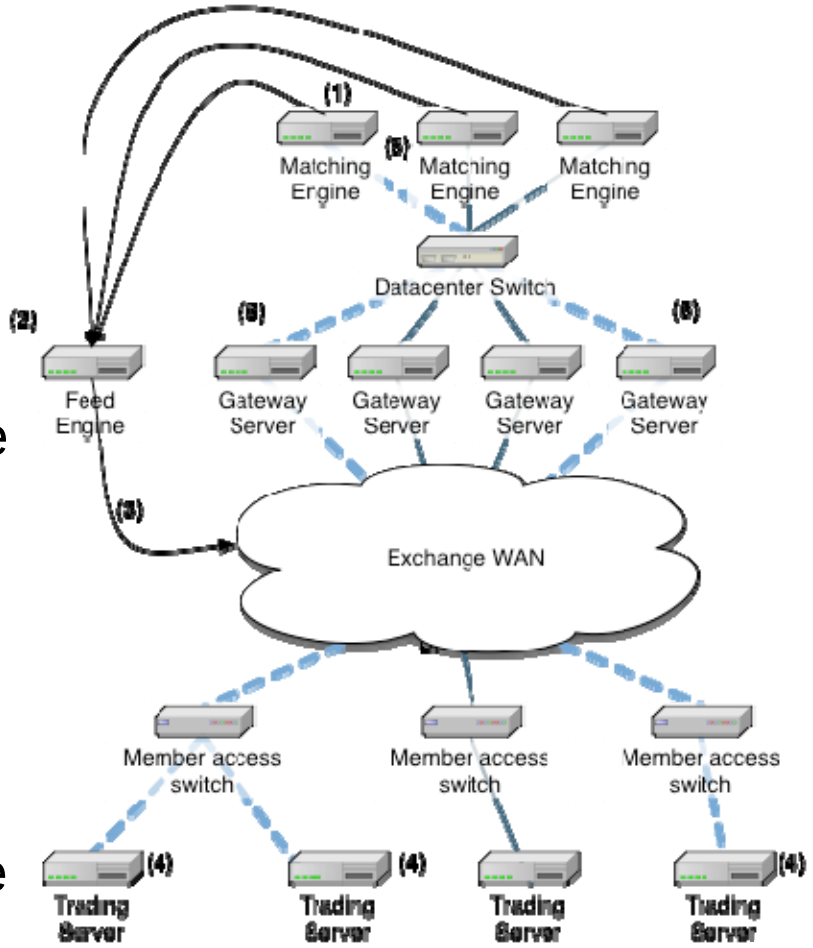
8.2.2011

Overview

- How does low latency trading work?
- Goals
- What is FAST?
- Solution - Why use HT and an FPGA?
- Implemented HW Structure
- Performance results
- Summary

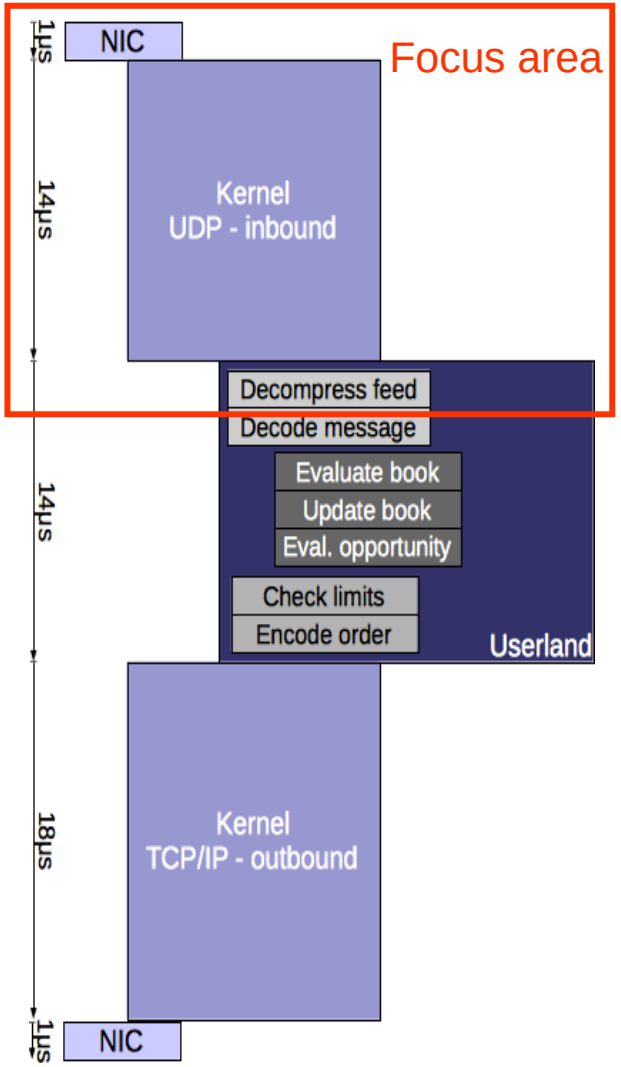
How does low latency trading work? (1)

1) An opportunity is created at the matching engine
2) The matching engine creates an update and sends it to the feed-engine
3) The feed-engine multicasts it to all the clients
4) The client machines evaluate the opportunity and respond with an order
5) The gateway receives the order and forwards it to the matching engine.
6) The matching engine matches the *first* arriving order against the created opportunity - a trade happens



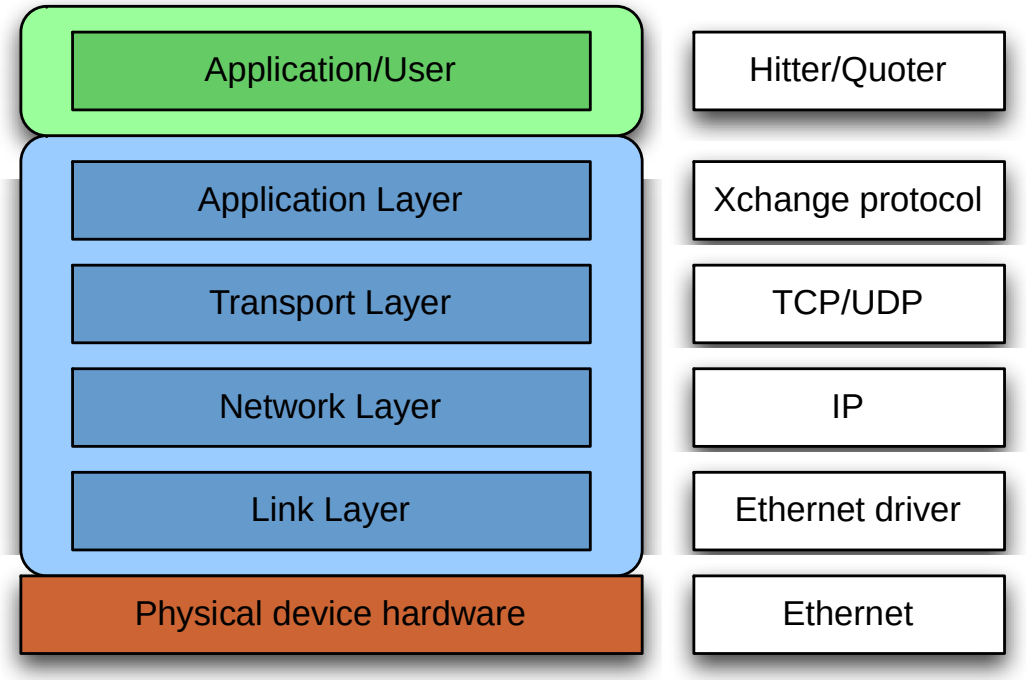
How does low latency trading work? (2)

- Current solution for traders:
  - Inbound traffic using standard NIC
  - Inbound kernel level IP-stack
  - User-level software for decoding, evaluation and trading
  - Outbound kernel level IP-stack
  - Outbound traffic using standard NIC
- Standard NICs are not designed for low latency communication
- Kernel level communication has a high latency
- Total latency of around 48µs



Goals

- Significantly reduce inbound latency
- Add an exchange protocol accelerator



What can be done to reduce latency?

- Use a faster network
  - Type of network is given by the exchange
  - Changing network speed incurs additional switch overhead
- Let the CPU run faster
  - Already done – cooling and reliability issues
  - Does not reduce NIC latency
- Use a GPU as accelerator
  - GPU is only good on parallel code
    but FAST is highly sequential dependent
  - High overhead for start-up – typically 1600 cycles to transfer data between CPU and GPU.

What can be done to reduce latency?

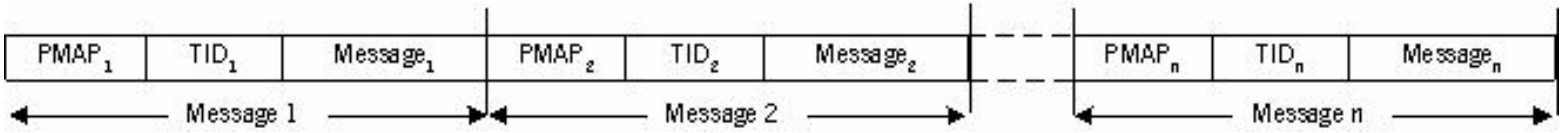
- Bypass Kernel
  - Good idea, therefore implemented in this project
- Use an FPGA as accelerator
  - NIC can directly be implemented in FPGA
  - Offload exchange protocol manipulation, normalize data
- Use a low latency host interface
  - HT has a lower latency compared to PCIe

What is FAST? (1)

- FAST stands for **F**IX **A**dapted for **St**reaming
  - Protocol to compress data streams for ‘efficient’ bandwidth usage
  - Trades CPU power for bandwidth – smart 10 years ago, now ...
- FIX is the **F**inancial **I**nformation e**X**change protocol (http://fixprotocol.org)
  - It is used to communicate trade related information
  - Industry driven standard formed by exchanges, banks, brokers

What is FAST? (2)

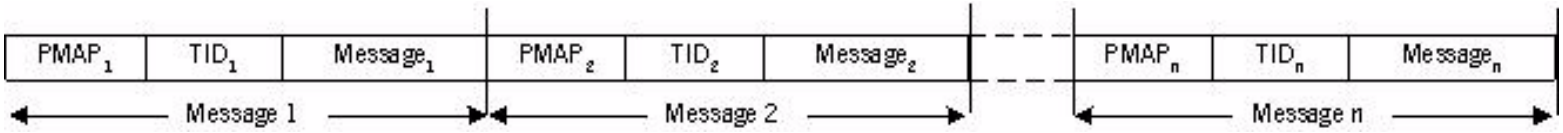
- Multiple messages in one UDP frame
  - NO size information in message
  - NO defined framing
- Templates specify messages
  - ALL used templates need to be known by receiver in order to be able to decode a stream
- Templates are a set of fields, sequences and groups
  - Groups: a set of fields
  - Sequences: groups that can occur multiple times



Enhanced Broadcast Solution message structure

What is FAST? (3)

- Each message begins with a Template ID (TID) and a presence map (PMap)
- TID is used to identify which template needs to be used for decoding
- PMap is used to specify which of the fields, groups or sequences are present in the current stream
  - PMap is a mask, there is one bit for every field
  - BUT: There are some exceptions – added complexity



Enhanced Broadcast Solution message structure

What is FAST? (4)

- There are several types of fields:
  - Signed integer 32 / 64 bit
  - Unsigned integer 32 / 64 bit
  - String (with an unlimited length)
  - Decimal (set of sint32 and sint64)

- Only bytes with relevant data are transmitted
  - Example: only 1 byte is sent for a sint64 with value ‘1’

- Only 7 bits of each transmitted byte are used for actual data, the 8th bit is used to mark the end of a field

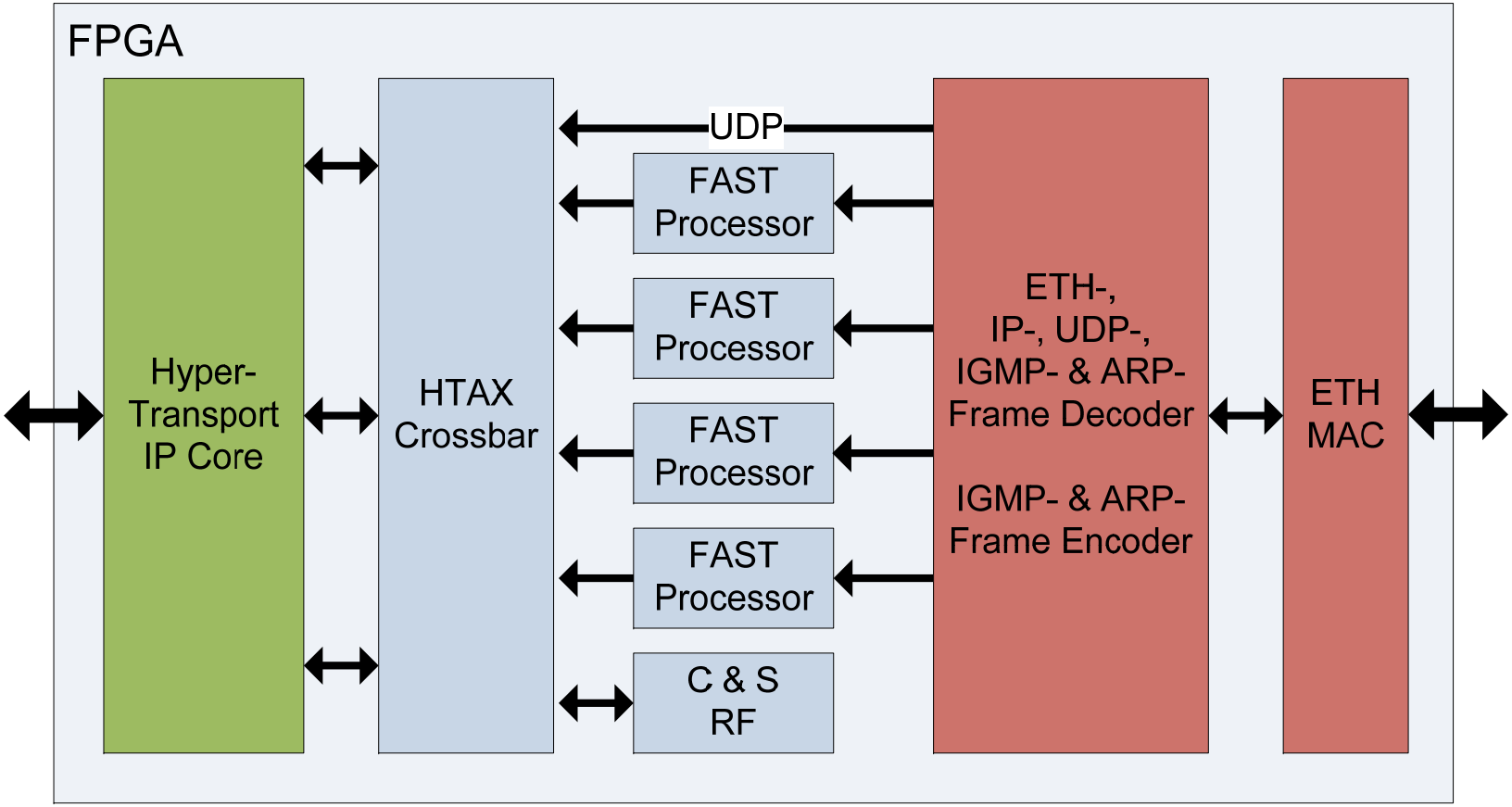
- Example :

	10000111_00101010_10111111	= 2 fields
First field:	1 byte, value: 00111111	= 0x63
second field:	2 byte, value: 00000011 10101010	= 0x3 0xA

Solution

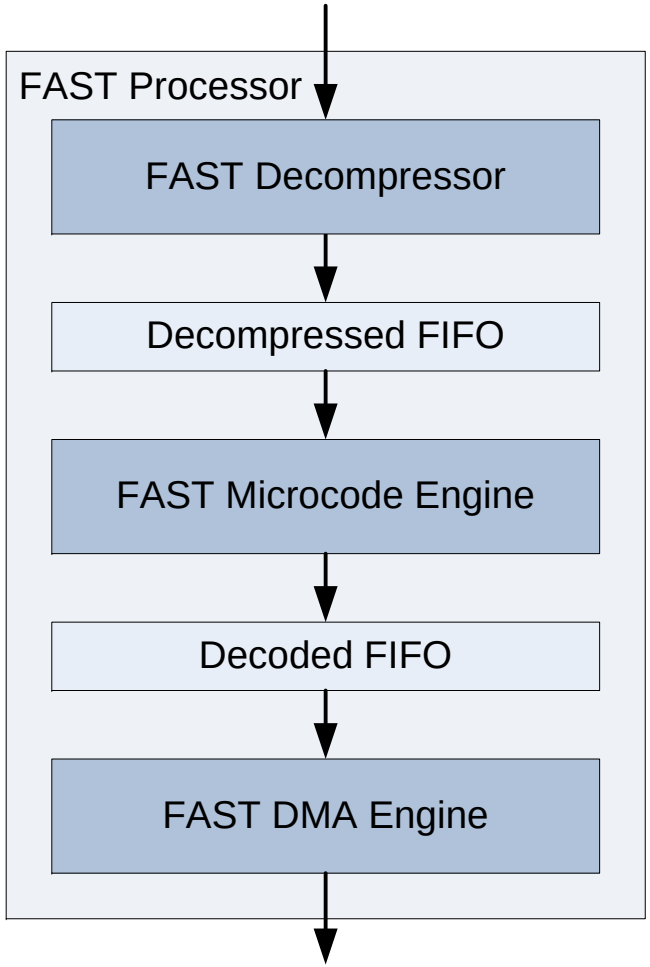
- Solution
  - FPGA with NIC
  - Offload as much as possible to the FPGA
    - Leverage strength of FPGA as a stream processor
  - Deliver normalized data to software on CPU
  - Use shared memory interface to share one incoming stream with multiple trading applications.

Hardware Structure (1)



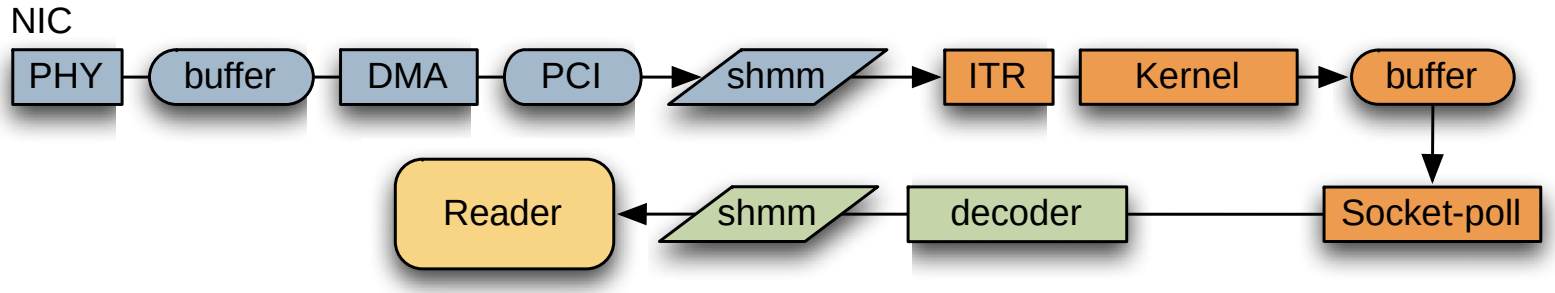
Hardware Structure (2)

- FAST Decompressor:
  - Looks at stop bits
  - Aligns all incoming fields to a multiple of 64 bit
- FAST Microcode Engine:
  - One program for each template
  - Decodes fields according to program
- FAST DMA Engine
  - Stores decoded templates in user address space (main memory)

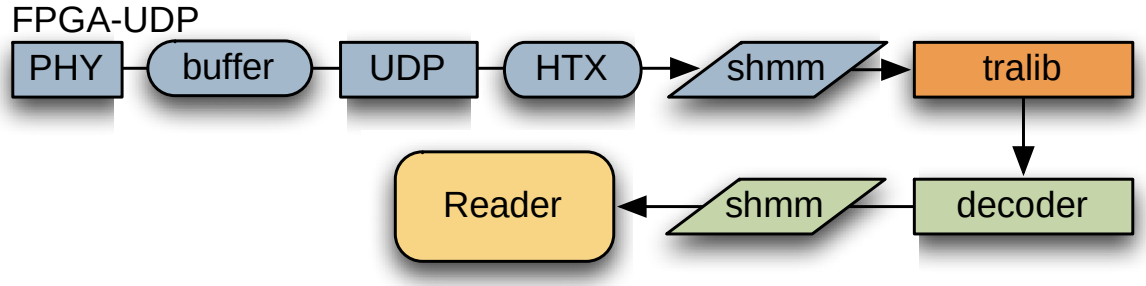


Performance results

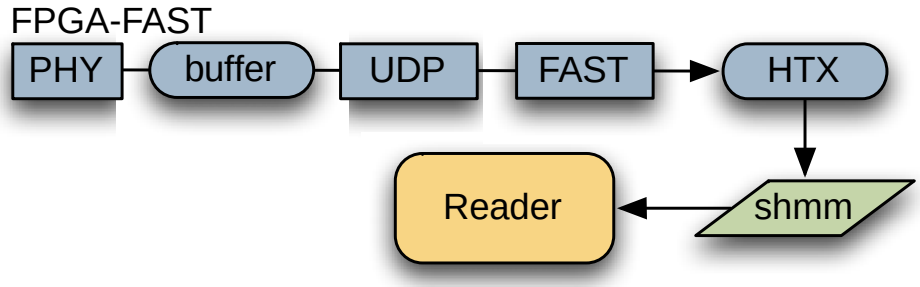
- 1:



- 2:



- 3:



Summary

- FPGA Accelerator proved to be feasible for efficient FAST protocol processing
  - Bypassing kernel IP stack provided most latency improvement
  - Implementing FAST handling further reduced latency
- HT interface provides a relevant building block for low latency trading

Thank You !

Any Questions ?