A Low Latency Library in FPGA for High-Frequency Trading

John W. Lockwood, Adwait Gupte, Nishit Mehta
Michaela Blott, Tom English, Kees Vissers

Conforto, Domenichetti, Faorlin. MAPD mod A 03-2021

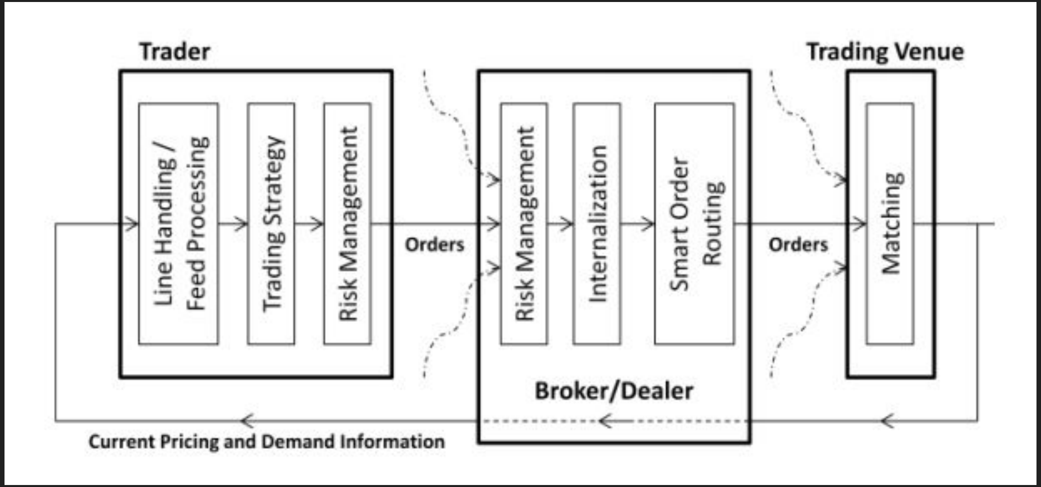
Outline

HFT ⇨ Rapid electronic trade in financial instruments

Current price info are **streams of electronic data**

Tiny gains on large numbers of transactions ⇨ Algorithmic Trading

Outline



Trader Decision maker (buy/sell)
Goals Risk management, **Return**

Broker Medium Trader - Venues
Goals **Direct** and **low-latency connection** from order to execution

Venues Matching supply and demand
Goals **Throughput** and **latency** (Earnings from matching fees)

Why Low-Latency?

- HFT traders exploit **inefficiencies** in the market

Market more and more efficient

The first players ordering are the ones able to profit from a given opportunity

- **Slippage** Change in the price after an order has been entered

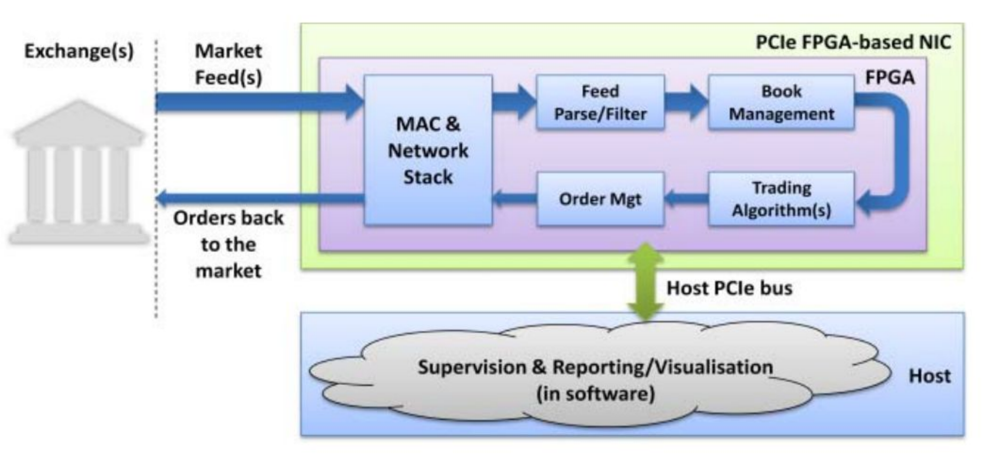
e.g: **ms reduction** in latency ⇨ increase arbitrage profitability by **$100M/year**

Software approach

Software: Pros high level programming, quick re-compile
Cons long & unpredictable response times (uncompetitive)

Bypassing the operating system’s kernel and networking stack reduces both latency and jitter
Claimed application-to-application latencies of 3.5μs for UDP and 4.0μs for TCP

FPGA-based **Smart NICs**

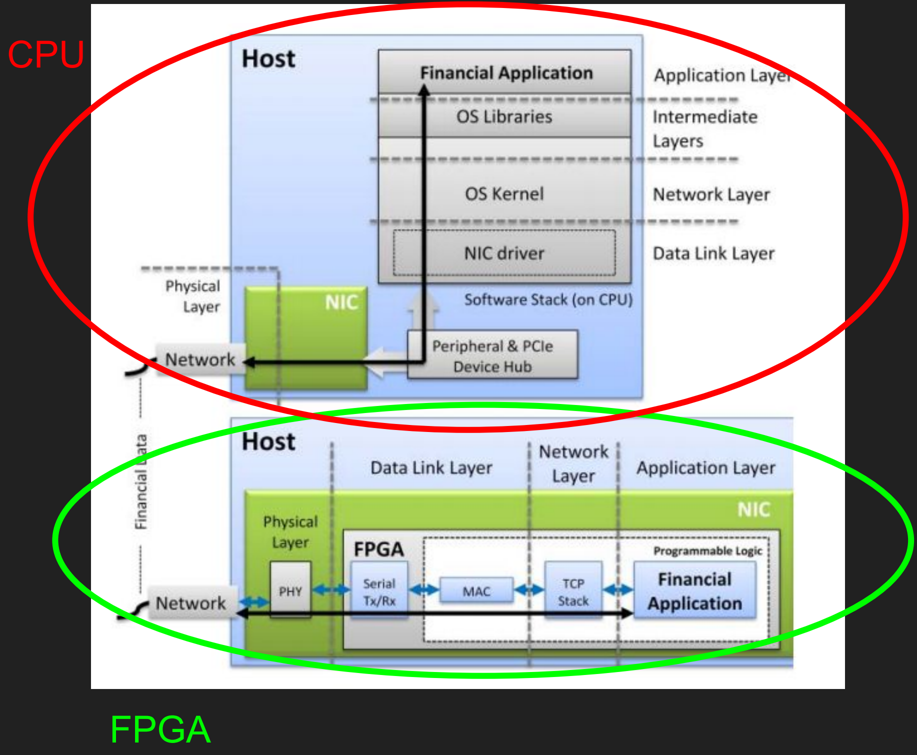


- **FPGA** implements the NIC controller
- **Bridge** between the host computer and the network
- **User-designed custom processing** logic to be integrated directly into the data path

Smart NIC: programmable trading platform supervised by a Host CPU

NIC trades turn-around: < **2µs**

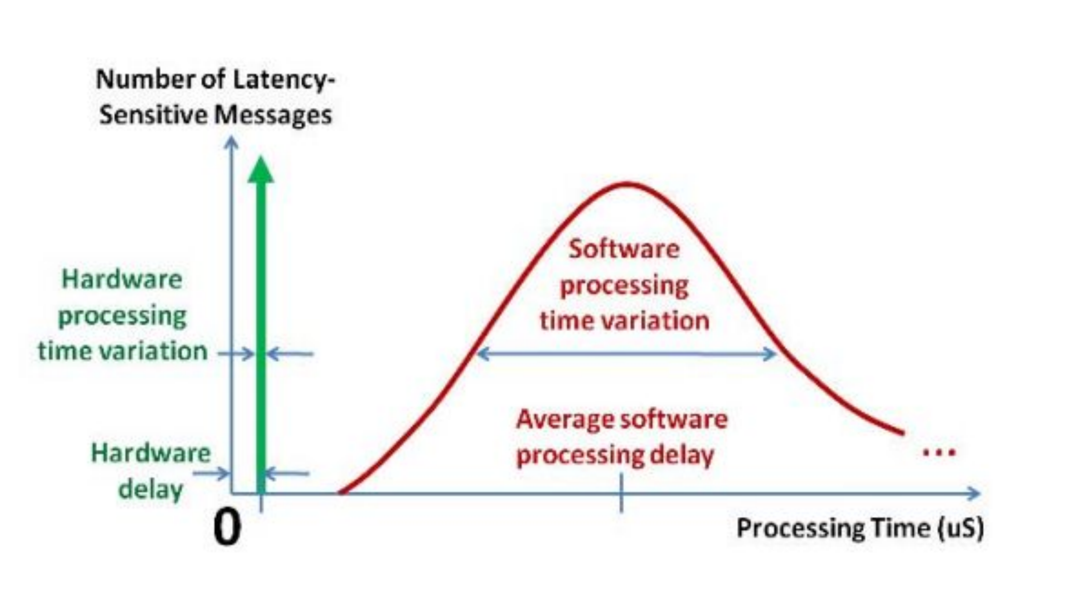
FPGA-based platforms



- During data transfer, **CPU is interrupted**
- **Higher latencies** - interrupt-driven stack, unpredictable PCIe transfers, cache misses
- **Ahmdal's Law** limits parallel processing

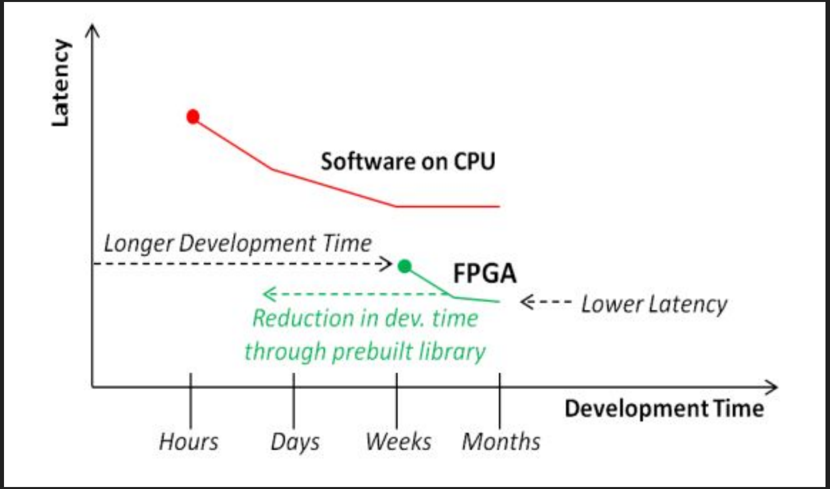
- Incoming data **directly fed** into custom-designed, highly-optimized, application-specific processing pipeline
- **Entirely avoided** latency and jitter from host OS
- Relevant packet information **extracted before the complete packet is received**
- Data available within the **same clock cycle** of its arrival
- Processing time **completely predictable** down to a clock cycle.
- Overall **Parallel Execution**

Processing time comparison


Number of Latency-Sensitive Messages
Hardware processing time variation
Hardware delay
0
Software processing time variation
Average software processing delay
...
Processing Time (uS)

- Processing time **completely predictable** down to a clock cycle

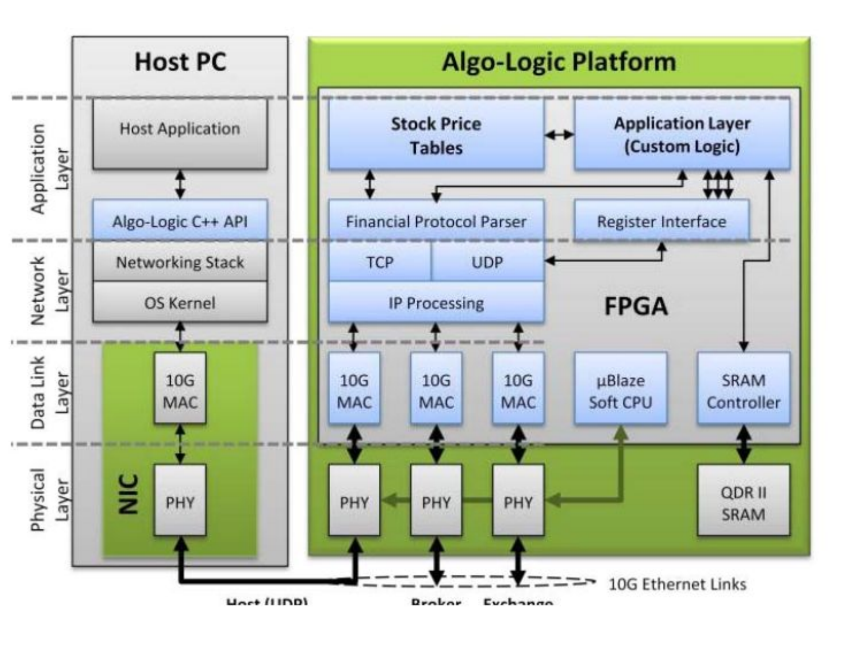
FPGA disadvantages



- **Great complexity** of FPGA development flow
- Financial developers are **not familiar** with FPGA
- Building and testing hardware **more time consuming** than software: lower abstraction level
- **Synthesis, placement and routing**: hours for a complex design. Software compilation - near-instant

Avoid HDLs: Include high-level programming environments
Library of **pre-built FPGA IP blocks** for networking and financial protocol parsing

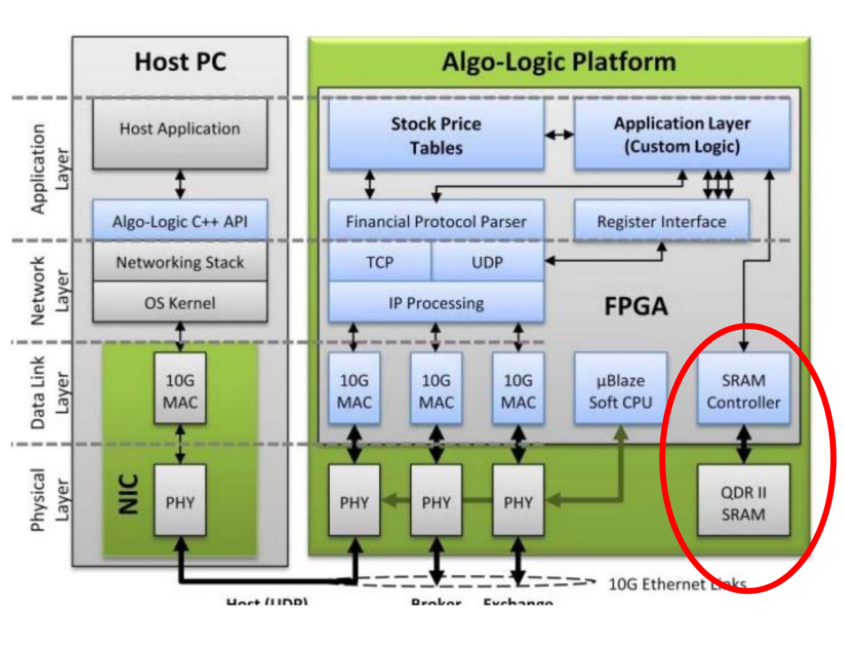
Auto-Logic Library



- Processing info as they pass through the FPGA
- Pre-verified IP blocks **understand and translate the exchange protocols**, focusing end-users on application development rather than interface components.

IP cores: network interface
External memories
Host software
Financial Processing Components: parsing and processing of standard and stock exchange protocols

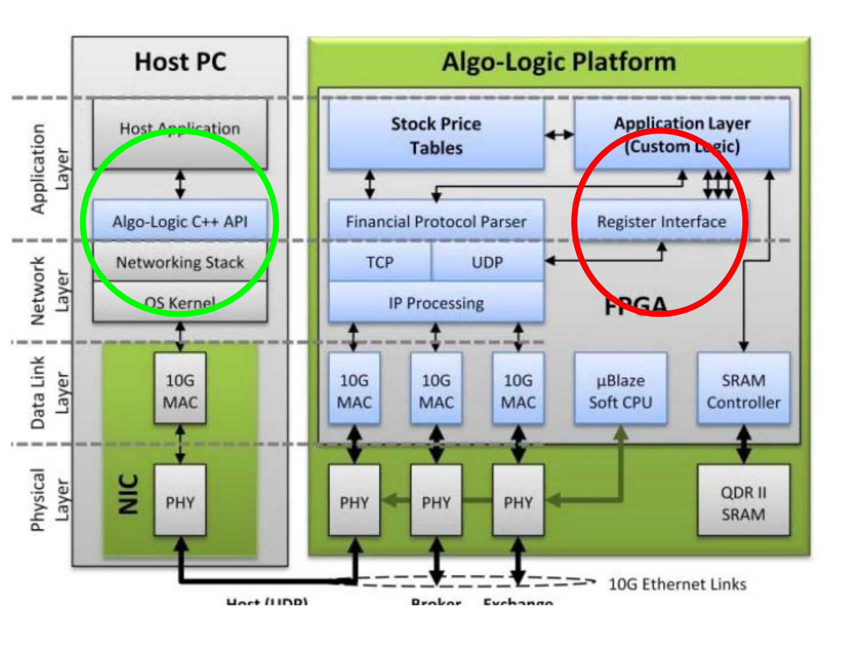
Auto-Logic Library



SRAM controller drives the on-board QuadDatarate (QDR) II Static RAM ICs.

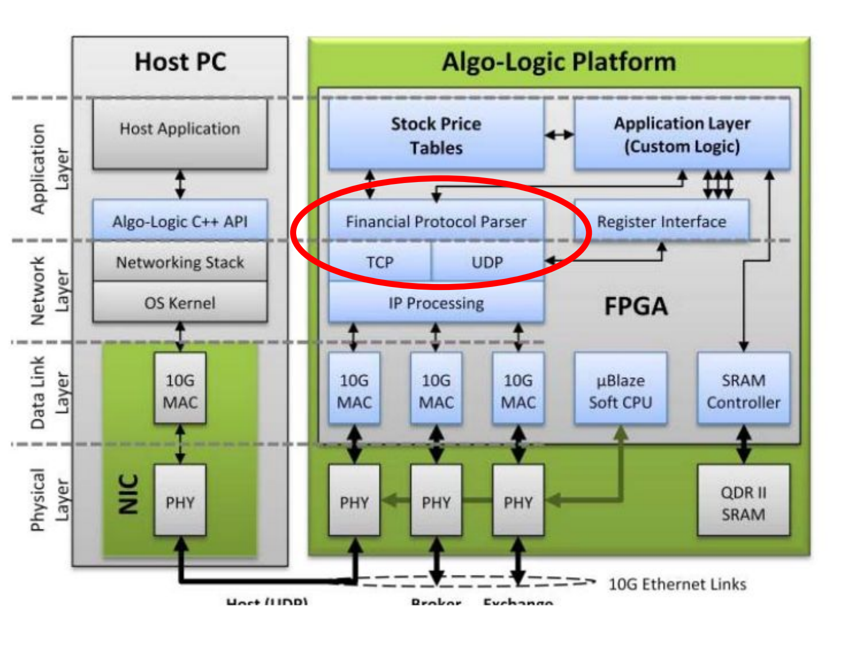
SRAM offers low latency, high-performance storage for small amounts of data

Auto-Logic Library



The **Register Interface** can be accessed via a **C++ API** on a host or controlled interactively via a web-based Graphical User Interface (GUI)

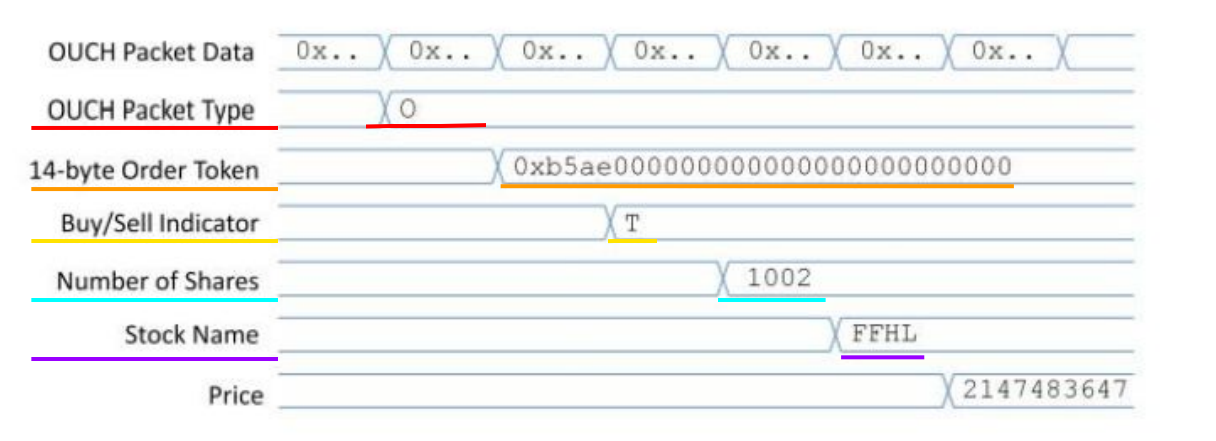
Auto-Logic Library



- The **Financial Protocol Parser** receives a **payload from the TCP/IP** Layer
- **Identifies message boundaries** for the data in the payload. Extracts individual fields and **raises a flag** when the value of each field becomes valid

This ensures that each field is extracted with the **lowest possible latency**

OUCH protocol (NASDAQ)



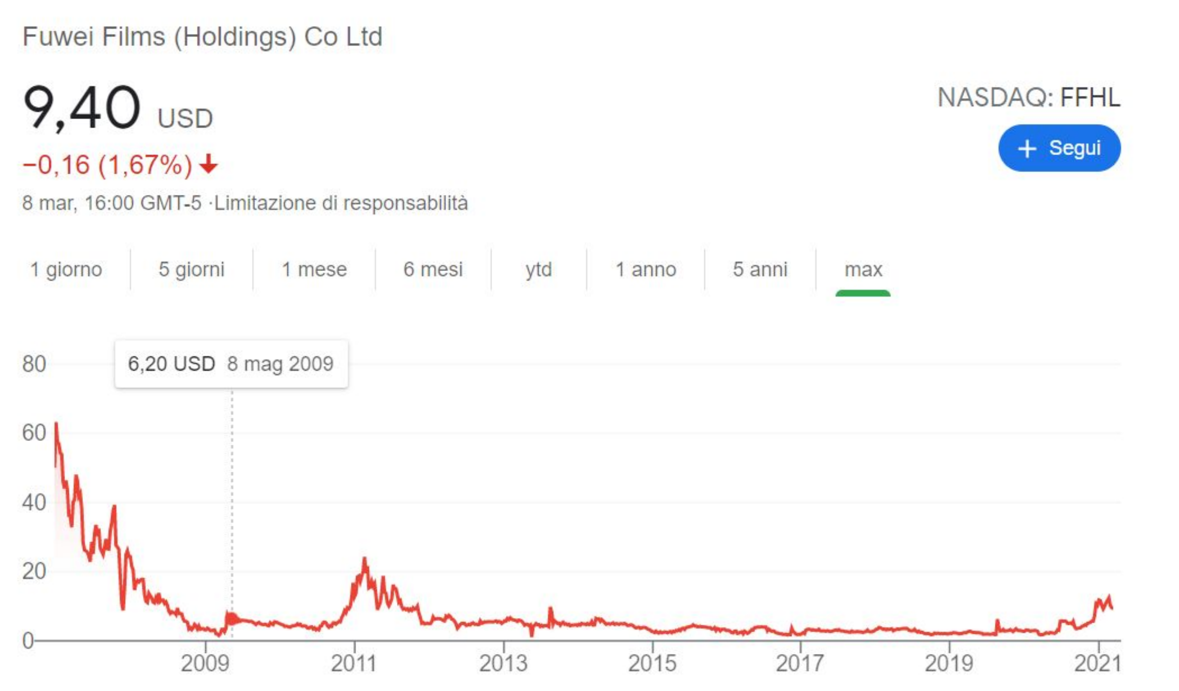
Message Type ('O', or 'Enter Order')
14-byte Order Token
Buy/Sell indicator ('T' for "Sell Short")
Number of Shares 1002
Stock Name (FFHL Fuwei Films (Holdings) Co., Ltd.)

OUCH Packet extraction time*

6.4*ns*

*(OoM faster than CPU extraction-transporting and full recieving before processing)

A working example..



Fuwei Films (Holdings) Co Ltd

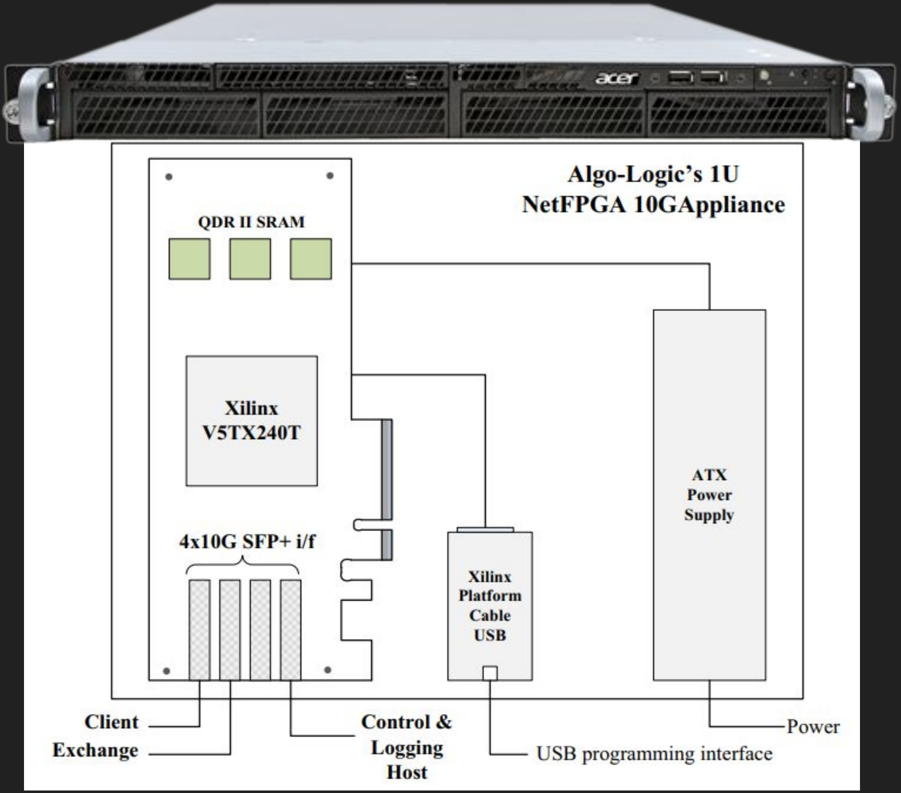
9,40 USD

−0,16 (1,67%)

8 mar, 16:00 GMT-5 ·Limitazione di responsabilità

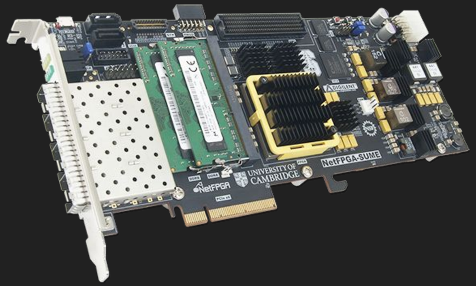
NASDAQ: FFHL

Real implementation

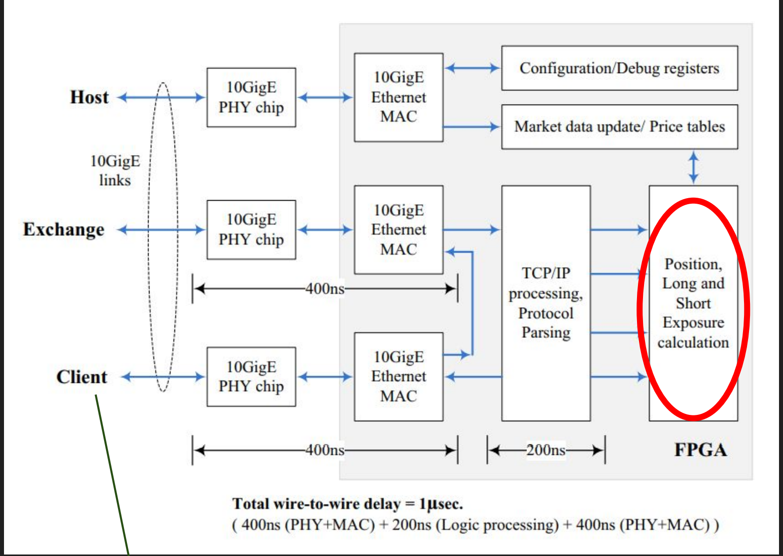


HFT Appliance using **Low Latency library** and the **NetFPGA-10G** card

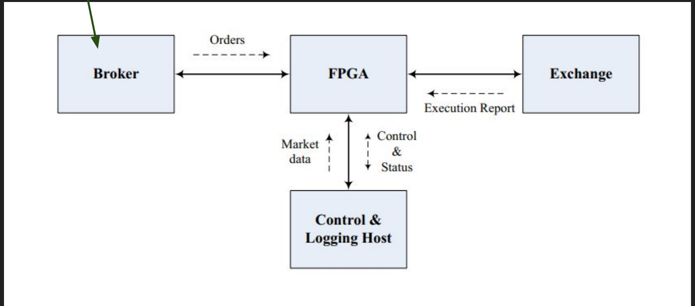
Must-know: price of instrument (security) ordered. Market data **fed into the card** via UDP/IP datagrams. Prices for **all 8000 securities** traded on U.S. exchanges fit within the FPGA's **on-chip memory.** (no PCIe!)



Real-time exposure tracking -



Total wire-to-wire delay = 1µsec.
(400ns (PHY+MAC) + 200ns (Logic processing) + 400ns (PHY+MAC))



- Critical to maintain a **margin of safety for brokers**

- **"Bump in the wire"** between broker and exchange;
- Processing FIX execution reports coming from the exchange while passing on orders coming from the involved brokers;

The Exposure and Position calculation module receives the traffic from the TCP/IP module and updates the market data/price tables.
FIX execution reports are forwarded to the clients providing details of order execution

Current market prices are received via UDP
Prices are needed to calculate exposure

Results

Total delay can be reduced significantly by performing **all operations in the FPGA.**

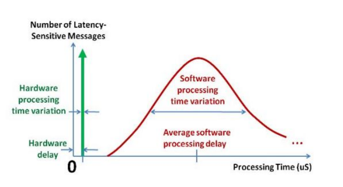
Rather than reading data across a memory bus, **data is stored in fast on-chip memory.**

with further improvements

best case scenario

wire-to-wire latency: 1μs with **6ns** of jitter against **5μs** and **600ns**

which phase of the *clk* I arrive (156MHz)



round-trip latency through this design is between **one and two orders of magnitude lower than software** implementation (in the worst case scenario);

Application can be assembled out of existing library components: the Low-Latency Library components were **simply connected together** to form the infrastructure.

Conclusions

- Time required to transfer a message from the network to the CPU then back to the network takes a **few microseconds.**

- Smart NICs go one step further**, offloading processing to a FPGA** on the network adapter.

Software	**Hardware**
Short development time	Processing time: clock cycle
High Latency and jitter	Longer development time

Auto-Logic's Library

Infrastructure components, domain-specific gateware;
Prices Storing: **on-chip memory;**
Application **track exposure and positions** of multiple traders;
End-to-end **stable latency**: 1μs (2 OoM lower than software).

LAB Report

Dizionario: IP, Internet Protocol , trasmissione di dati online, no controllo di flusso/errore (livello 3)
TCP: protocollo di trasporto di livello superiore, con controllo di errore (livello 4)
Jitter si riferisce invece alla variazione statistica nel ritardo di ricezione dei pacchetti trasmessi, causata dalle code interne ai router congestionati.
UDP: usato in combinazione con IP
NIC: scheda di rete (network interface controller)

ASIC Application specific integrated circuit - Wikipedia

Pendulo, Pandulo, Rotula, Mellino, Illice, Brontolo