A Low-Latency Library in FPGA Hardware for High-Frequency Trading (HFT)

John W. Lockwood, Adwait Gupte, Nishit Mehta

Algo-Logic Systems Inc.
Santa Clara, CA
{jwlockwd, adwait, nishit}@algo-logic.com

Michaela Blott, Tom English, Kees Vissers

Xilinx Inc.
Dublin, Ireland, San Jose, CA
{michaela.blott, tom.english, kees.vissers}@xilinx.com

Abstract—**Current High-Frequency Trading (HFT) platforms are typically implemented in software on computers with high-performance network adapters. The high and unpredictable latency of these systems has led the trading world to explore alternative "hybrid" architectures with hardware acceleration. In this paper, we survey existing solutions and describe how FPGAs are being used in electronic trading to approach the goal of zero latency. We present an FPGA IP library which implements networking, I/O, memory interfaces and financial protocol parsers. The library provides pre-built infrastructure which accelerates the development and verification of new financial applications. We have developed an example financial application using the IP library on a custom 1U FPGA appliance. The application sustains 10Gb/s Ethernet line rate with a fixed end-to-end latency of 1μs – up to two orders of magnitude lower than comparable software implementations.**

Keywords- Algorithmic, trading, latency, HFT, FPGA

I. INTRODUCTION

High-Frequency Trading (HFT) refers to rapid electronic trade in financial instruments. Trading venues make current pricing information available as continuous streams of electronic data. Traders monitor these streams, reconstructing pricing and demand for relevant stocks, options, futures and currencies to determine when and what to trade. Buy/Sell orders are then sent to the exchange as soon as possible. Algorithms take advantage of fleeting variations in stock price or demand over time or between different exchanges, accumulating profit by making tiny gains on large numbers of transactions. Traders avoid holding significant overnight positions, with individual stocks being held only for a few seconds before being re-sold [23]. HFT is growing rapidly and is estimated to have accounted for more than 70% of all trades made on US equity markets during 2010 [12].
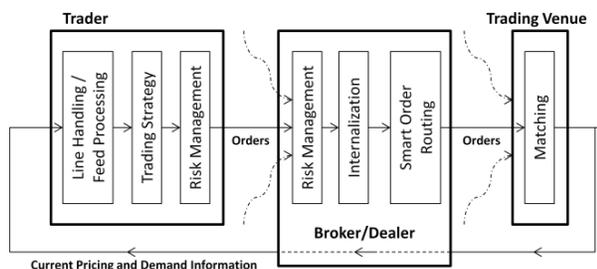

Figure 1   traders, brokers and trading venues

A. Traders

Traders receive multiple high-datarate UDP/IP market data streams over the network from trading venues. An initial *line handling* stage uses redundancy in the streams to replace any information temporarily missed due to packet losses in order to reconstruct the necessary parts of the *order book* of the exchange. Figure 1 illustrates how traders, brokers/dealers and trading venues (*exchanges*) interact in the HFT ecosystem.

Traders employ a range of algorithms and heuristics to decide when and what to trade. These approaches combine current market data with historical data and outputs from computational models. Trading decisions may be simple and quick, slow but smart, or (preferably) quick and smart. Quantitative Analysts ("Quants") are employed to develop the most successful and repeatable approaches.

The final step is to enter these orders into the exchange –via a broker. Further computation is used to ensure that the trader's exposure is carefully managed.

B. Broker/Dealer

The function of a Broker in the trading infrastructure is to be a market participant who takes orders from clients (traders) and passes them to the exchange. Brokers manage risk according to regulatory requirements [1]. Providing the most direct and low-latency connection possible to an exchange inherently makes a broker more attractive to a client.

Many financial instruments are traded at more than one venue. Each venue charges some small *matching fee* per transaction. Accordingly, once an order is deemed compliant with regulations, the optimum execution venue must be chosen to ensure both the fastest execution and the lowest transaction fees. This step, known as *order routing*, is one way for brokers to increase their profits without compromising the level of service to clients. Another way by which brokers enhance profitability is by matching orders from different clients with each other internally, without sending them to the exchange. This *internalization* follows strict regulations.

Stock exchanges handle a large amount of orders from different brokers. Incoming bids are *matched* with *ask prices* set by current stock owners, triggering trades. The exchange is one of the few links in the infrastructure where both throughput and latency are critically important. The faster an exchange is able to match orders, the more efficient and attractive it becomes to traders who care about latency. Also, the greater the number of transactions handled by the exchange, the more matching fees it can earn.

9

C. Why Low Latency is Important in HFT

HFT traders attempt to exploit fleeting inefficiencies in the market. By doing so, they not only make money but also make the markets more efficient. However, due to the increase in number of players in the HFT space which try to exploit these opportunities, the first few players to execute orders may be the only ones able to profit from a given opportunity. This is a first reason why latency matters.

Another reason is the phenomenon of slippage which refers to the price of a highly liquid security moving away from its expected price after a market order for that security has been entered. The longer the time between a trader making a decision to trade and the order reaching the execution venue, the greater is the possibility and magnitude of slippage which translates into cost for traders.

Finally, some HFT strategies such as latency arbitrage depend upon the ability to access market data and execute orders faster than other investors. Profiting from this activity requires traders to be able to react in real time to market events using both low-latency and low-jitter market access. Every millisecond reduction in latency can improve arbitrage profitability by more than $100M a year [13].

HFT strategies therefore require low-latency market access both to receive market data as well as to transmit new orders.

D. Other Important Factors

Throughput and flexibility are additional important considerations for HFT. Throughput is becoming more and more critical as HFT inherently involves large volumes of orders arriving at trading venues continuously for processing. In the trade of options and futures, for example, the volume of market data being propagated has increased exponentially. Flexibility is also vital to enable trading platforms to adapt to changing market conditions and trading strategies.

II. Survey of Current platforms for HFT

Most traders and brokers today implement their HFT platforms in software on commodity servers. This allows algorithms to be expressed in familiar high-level programming languages and be re-compiled quickly to make improvements. However, the industry-wide race to reduce latency is making the long and unpredictable response times of software systems increasingly uncompetitive. This section surveys software platforms for HFT, contrasting them with hardware-based approaches offering lower, more predictable latency.

A. Software-based HFT Platforms

Software providers such as Mantara, Ullink and QuantHouse offer customizable trading software packages designed for minimum latency. Much of the remaining latency in software-based trading platforms is due to the computer's operating system networking stack. To alleviate this, end users can combine trading software with specialized, low-latency network interface cards (for example available from Solarflare and Myricom) that accelerate parts of the networking stack in hardware and bypass the operating system's kernel. These techniques can significantly reduce overall latency compared to a naïve approach.

Several prior studies have been conducted into the latency of software-based approaches with low-latency Ethernet NICs. Measurements on a Linux server in [11] estimate a half round trip of 15-20μsec for packets through the operating system kernel and network stack. Lobo et al. [9] claim a roughly 40μsec round trip time for un-optimized systems, while transmit and receive latencies of 2.9μs and 6μs were measured when a TCP offload engine was used. It is important to remember that none of these figures takes into account the additional latency of the end-user application. Myricom's low-latency 10G Ethernet NIC products offer commercial software called Datagram Bypass Layer (DBL), which allows high-priority user threads to send and receive IP frames directly to and from the NIC, bypassing the operating system's kernel and networking stack. This reduces both latency and jitter, resulting in claimed application-to-application latencies of 3.5μs for UDP and 4.0μs for TCP [6]. A software decoder for the OPRA option pricing feed on an Intel Xeon CPU processed 3 million OPRA messages per second per CPU core with a round-trip latency of 4μs using Myricom's 10G Ethernet adapters and DBL kernel bypass software [24].

Infiniband-based networking can reduce latency significantly compared to Ethernet. For example, Qlogic's 7300-series Infiniband NICs claim HPC Message-Passing Interface (MPI) network latencies of as low as 1μs [7]. (his figure excludes latency at the application layer.) Despite Infiniband's latency advantage, Ethernet is currently more prevalent in the HFT ecosystem. 40Gb/s and 100Gb/s Ethernet devices are already on the market, suggesting Ethernet will continue to dominate for the foreseeable future.

B. Custom Hardware-based HFT Platforms

The comparatively large and unpredictable latency of software-based HFT platforms has led the industry to explore alternative lower-latency approaches using custom hardware. ASICs are typically not considered for HFT as they lack the flexibility to be reconfigured to handle new protocols. GPUs are optimized for throughput and cannot offer sufficiently low latency due to their deep pipelines. Field-Programmable Gate Arrays (FPGAs) offer the performance of custom hardware without compromising on flexibility or latency.

FPGAs can be used in a number of ways to accelerate financial applications. One approach, referred to as Hybrid Computing, is used for example in risk management, option pricing and portfolio modelling and can accelerate performance by three orders of magnitude while reducing energy costs [25]. Hybrid computing blends traditional multi-core CPUs with FPGA-based co-processors for acceleration. CPUs integrate several powerful processing cores and large multi-level caches and are best suited to control-intensive parts of an application. FPGAs perform best at non-floating-point tasks such as integer, binary, character or fixed-point data processing. FPGAs also offer the possibility of tailoring the accelerator exactly to the application for optimal performance and efficiency.

In a typical Hybrid Computer, the CPU is connected using a high-bandwidth interconnect such as FrontSide Bus (FSB), PCI Express or QPI to one or more FPGA modules. High-level programming tools enable applications to be compiled seamlessly to both the CPU and FPGA. Vendors such as

Maxeler Technologies and Convey Computer (amongst others) offer such platforms. Examples of Hybrid Computers specifically targeted at HFT include Nomura's NXT Direct and Deutsche Bank's Autobahn Ultra. Both products were developed to accelerate pre-trade SEC regulatory compliance and risk management checks on orders.
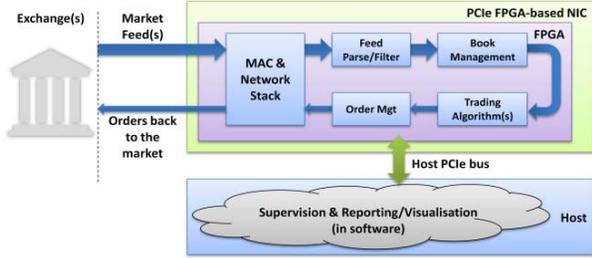

Figure 2   Typical Smart NIC in a supervised financial trading application

FPGA-based "Smart NICs" offer another way to apply programmable logic to the acceleration of financial applications. Smart NICs typically bring together high-speed network interfaces, a PCIe host interface, memory and an FPGA. The FPGA implements the NIC controller, acting as the bridge between the host computer and the network and allows user-designed custom processing logic to be integrated directly into the data path. As illustrated in Figure 2, this allows a smart NIC to function as a programmable trading platform under the supervision of a host CPU.

Many vendors offer FPGA-based "smart NICs". A selection of products targeted specifically at financial trading appears in Table I. The NIC hardware is typically accompanied by programmable logic IP implementing networking-related functions as well as a PCIe host interface, with or without direct memory access (DMA). Some products include a very limited set of finance-specific IP blocks.

Table I   SMART NICS FOR LOW-LATENCY FINANCE

Product	Interfaces	Platform IP
DINI group DNPCIe_10G_HXT_LL	1x10GE SFP+, 1xIB CX4 PCIe Gen2 x4	Low-level PCIe driver, 10G MAC, TCP offload engine, DDR memory controller, PCIe DMA engine and FIX parser
Advanced IO V5021/V5022	4x10GE SFP+ PCIe Gen1 x8	PCIe driver, *expressXG* SDK and basic network/host I/O IP blocks
Accelize XPS4S530LP-20G/ XPS4S1050GT-20G/ XPS4S1050GT-10G	1x/2x10GE SFP/SFP+ PCIe Gen2 x4/x8	PCIe driver, *HCE* SDK, 10G MAC, multi-session TCP offload engine, DDR memory controller

For these products, precise latency figures are application-dependent and difficult to obtain. However, Accelize claims their smart NICs can turn around financial trades on the FPGA in under 2µs. This is significantly lower than the minimum latency achievable with software, as discussed earlier.

Within the research community, the original NetFPGA [2][3] was used as base platform for building FPGA-accelerated network processing applications. It was developed by Stanford University in collaboration with Xilinx and other partners. In [4], a latency measurement circuit is built on the NetFPGA to monitor the distribution of the difference in arrival times between market data on a pair of redundant ("A/B") feeds. An event processor for algorithmic trading on NetFPGA in [10] reduces latency up to two orders of magnitude compared to software. Morris et al. present an FPGA-assisted HFT engine implemented on Celoxica's AMDC card which improves message throughput by 12 times, while reducing latency [11]. An FPGA-based feed handler for the NASDAQ ITCH protocol achieves a low, predictable end-to-end latency of 2.7µs [26]. A comparable software implementation (with no kernel bypass) exhibits a typical latency of 38µs, varying more than ±20µs.

Finally, some financial processing systems are completely implemented on FPGA-based platforms without any host CPU yielding best possible latencies. As oncoming traffic no longer needs to make a round trip over the system's PCIe interface for processing in the CPU, the inherent latency and jitter introduced by the host's operating system are entirely avoided. The approach presented in this paper fits into this category.

III.   ADVANTAGES AND DISADVANTAGES OF FPGAS FOR LOW-LATENCY FINANCE

A. Advantages

Modern FPGAs can implement most aspects of any HFT application. Incoming market data can typically be processed completely on the FPGA without needing to travel over and back to a host CPU. Figure 3 compares an FPGA-based platform to a traditional software trading platform.
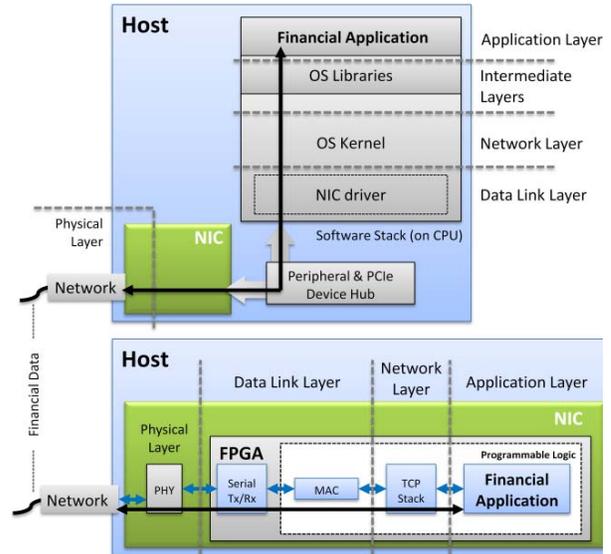

Figure 3   Software-based (top) and FPGA-based (bottom) trading platforms

In comparison to the FPGA-based trading platform, software-based platforms require network traffic to be received

on a network adapter. Relevant traffic is transferred to memory and the CPU is interrupted to handle the application processing. After this processing is complete, the data is transferred back to the network adapter and transmitted over the network. The interrupt-driven software stack, unpredictable PCIe transfers and cache misses make network latencies higher and less predictable in software implementations. By contrast, in the FPGA implementation, the incoming network data is fed directly into a custom-designed, highly-optimized and application-specific processing pipeline via hardware PHY and MAC blocks.

Furthermore, relevant information within a packet can in fact be extracted before the complete packet is received. The data is directly available within the same clock cycle of its arrival time which is significantly different from any software implementation. (The network stack itself holds the complete packet at least once before its available for processing.) Directly related to this, it's important to highlight that there is literally no jitter in extracting and processing incoming data. The availability and the processing time for any piece of information is completely predictable down to a clock cycle. FPGAs can therefore naturally achieve significantly lower latencies with minimal jitter, as shown in Figure 4. This is one of the key advantages that FPGAs offer, as the example in Section V proves.
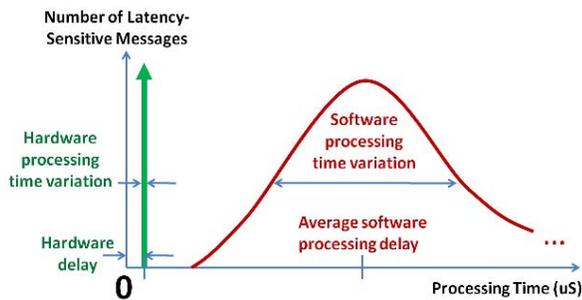

Figure 4   Throughput vs Latency in Software (red) and Hardware (green)

Along with lower latency and minimal jitter, FPGAs can achieve significantly higher throughput using parallelism. For example, a processing pipeline that implements a financial application can easily operate at 200MHz. Given a 256bit data path, throughput would reach 51Gb/s, corresponding to a message rate of 76Mpps. In addition, multiple pipelines can be instantiated in parallel when data dependencies do not exist. By comparison, multi-core CPUs do not necessarily improve performance due to the effects of Amdahl's Law. Inter-core and inter-thread signalling overheads are increased, often requiring complex management techniques in the application and in the OS. For example, Mantara advertises a maximum message rate of 10Mpps for their Expressway market data software, which is significantly lower than what can be achieved on an FPGA.

Although in general the low- abstraction level in the design entry for FPGAs increases the design complexity, it also brings advantages. The available bit-level access to incoming data greatly simplifies protocol parsing which is a task that is tedious to describe with a standard programming language.

Finally, in contrast to ASICs, FPGAs offer the performance benefits of custom hardware while retaining programmability. Trading algorithms can be continually improved or circuit bugs fixed in the field.

B. Disadvantages

FPGAs also have disadvantages compared to traditional software-based approaches. At the root of this is the greater complexity of the FPGA development flow. Firstly, many developers of financial applications are unfamiliar with FPGA technology generally and lack the expertise to use the hardware-oriented FPGA development tools in particular. Secondly, building and verifying new hardware is more time-consuming than writing new software due to the significantly lower abstraction level in the design flow. However, the FPGA delivers better throughput and lower latency than a software implementation will ever be able to yield in return for this time investment. This is conceptually illustrated in Figure 5.
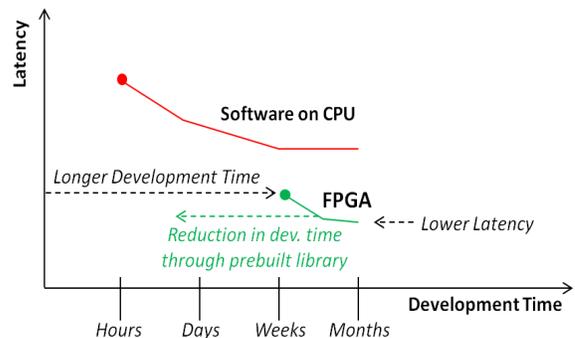

Figure 5   Latency vs Development Time for Software and FPGA

To overcome this problem, some FPGA-based HFT products include domain-specific high-level programming environments, avoiding the need for designs to be described in hardware description languages (HDLs). Within the research community, a more general "query compiler" has been introduced in which HFT application primitives can be efficiently expressed [22]. We address this problem by providing a library of pre-built FPGA IP blocks for networking and financial protocol parsing. This allows end users to focus on their application without first constructing basic infrastructure. This approach dramatically reduces development time, as is shown in Figure 5 and discussed in the example presented in Section V.

The last stages in the FPGA design process relates to synthesis, placement and routing which can still take hours for a complex design whereas software designs compile near-instant in comparison. This however is acceptable as trading strategies are relatively long-lived – a typical algorithm may have a shelf life of several days [19] while an FPGA implementation time takes hours in comparison.

IV. Algo-Logic's Low Latency Library and FPGA Implementation on NetFPGA-10G

Algo-Logic's Low-Latency Library is gateware that is compatible with standard FPGA hardware platforms. One supported platform is the NetFPGA-10G [8] which is a quad-port, 10-Gigabit Ethernet successor to the original NetFPGA card, based on a Xilinx Virtex-5 device.

Algo-Logic's Low-Latency Library processes financial protocols used by trading venues in the US, Europe and other regions, extracting information from the packets as they flow through the FPGA. The library components are used to construct custom trading applications, reducing time-to-market.

Algo-Logic's library of low-latency FPGA IP blocks (shown in Figure 6) can be divided in two main categories. A set of *Infrastructure Components* includes generic IP cores providing interfaces for the network, external memories and host software. *Financial Processing Components* parse and process standard and stock-exchange-specific protocols. The next sections describe these blocks in more detail.
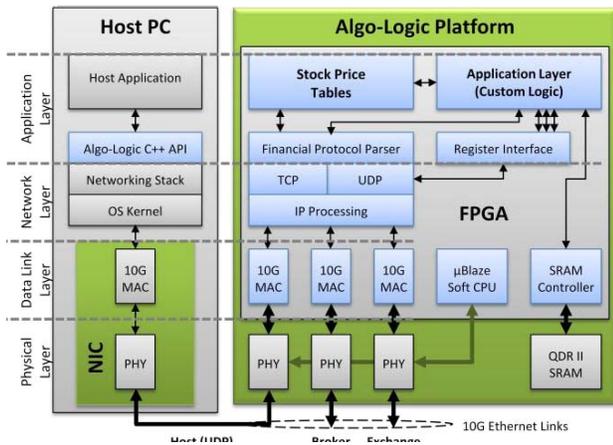

Figure 6   Algo-Logic's Low-Latency FPGA IP Blocks

A. Infrastructure Components

All components share a common, standardized interface with a 64-bit (8-byte) data word and protocol conforming to the industry-standard AXI-4 Stream specifications [20].

An SRAM controller IP block drives the on-board Quad-Datarate (QDR) II Static RAM ICs [5]. SRAM offers low-latency, high-performance storage for small amounts of data.

A Register Interface module controls and monitors the status of registers written and read by host software. The Register Interface contains multiple register types, including write-only Configuration Registers, read-only Status Registers, general-purpose read/write registers and write-only registers. Configuration and status update words are transmitted and received via UDP.

The Register Interface can be accessed via a C++ API on a host or controlled interactively via a web-based Graphical User Interface (GUI). The UDP-based interface allows the FPGA to be controlled and monitored from any host on the network.

The TCP/IP Processing Layer separates the headers from the payloads, stores the headers and forwards the payloads to the application layer for further processing.

The Application Layer processes packet payloads and sends them back to the TCP/IP Layer. Packets are reassembled with their corresponding headers and sent out to the Ethernet MAC. A new TCP/IP checksum is computed over the resulting payload after application-level processing is complete.

Finally, a 10G Ethernet MAC IP block from Xilinx provides the interface to the NetFPGA-10G's on-board discrete PHY ICs. A MicroBlaze soft processor core is used to initialize the on-board PHYs. Note that the soft core is **not** used in the datapath to process time-critical packets. However, if required, it could be used to run a simple software application which could communicate with the rest of the system through the Register Interface.

B. Financial Processing Components

Algo-Logic's library includes IP blocks designed to process application layer messages for HFT. These messages consist largely of the orders and order execution reports sent between clients, brokers and exchanges. A set of pre-verified IP blocks understands and translates the exchange protocols, enabling end-users to focus on application development rather than interface components. Each component is discussed in more detail below.

1) Financial Protocol Parser

The Financial Protocol Parser receives a payload from the TCP/IP Processing Layer and identifies message boundaries for the data in the payload. The parser 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. Algo-Logic's library currently has parsers for ASCII protocols such as Financial Information eXchange (FIX) and binary protocols such as OUCH for NASDAQ [14], XPRS for DirectEdge [15], ArcaDirect for Arca [16], and Native Trading Gateway for London Stock Exchange (LSE) [17]. Support for BATS BOE [18] will be added shortly.

An example of how the Finance Protocol Parser in Algo-Logic's IP library parses a message with minimal latency is shown in the waveform in Figure 7. An incoming OUCH message packet is presented to the Financial Protocol Parser 8 bytes-per-cycle as streams through the 64-bit AXI-Stream data path. As each field is extracted, the value is transferred to internal I/O pins and a valid flag is raised.

The block extracts the OUCH message type ("O", or "Enter Order") and the 14-byte Order Token. The Buy/Sell Indicator field is "T" ("Sell Short"). The number of shares is "1002", with the stock symbol identified as "FFHL". The remaining fields specify the asking price and specify that the order is not eligible for intermarket sweep.
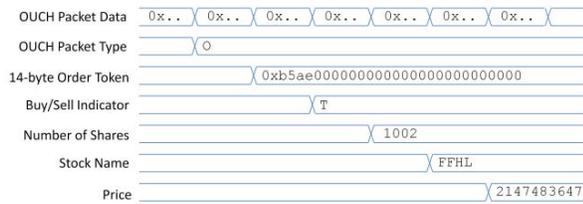
Figure 7   OUCH Protocol Parsing

The parser adapts automatically to variable-length data fields in the packet. With a 156 MHz clock and a single set of Delay Flip/Flops (DFFs), the circuit extracts all OUCH packet fields within exactly 6.4 nanoseconds of the packet's arrival with no jitter. This is orders of magnitude faster than a similar extraction in software which not only incurs system-level delays for transporting the packet to the CPU, but also needs to receive the full packet before it can start parsing.

2) Market data parsing and on-chip storage for price data

In financial processing, applications need to know the price of instrument (security) that appears in the orders. To track prices, Market data is fed into the card via UDP/IP datagrams. The FPGA extracts price updates and stores this data to memory for each symbol. Prices for all 8000 securities traded on U.S. Exchanges fit within the FPGA's on-chip memory. The table size is scalable to support more symbols as required.
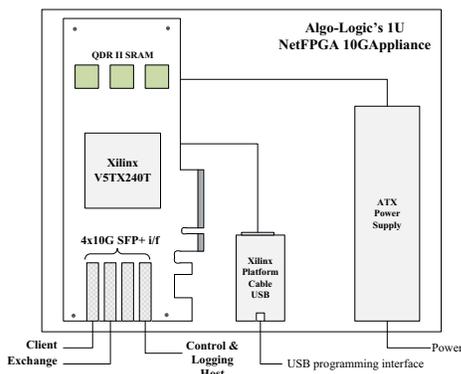

Figure 8   Algo-Logic's HFT Appliance

C. FPGA implementation on NetFPGA-10G

Algo-Logic has created a 1U rack-mounted HFT Appliance using the Low-Latency Library gateware and the NetFPGA-10G card. The hardware components of the appliance are illustrated in Figure 8. The appliance supports up to four 10G Ethernet interfaces which can be used to connect to the Exchange, a trader/broker and an optional host computer.

The next section illustrates how a high-performance, low-latency financial application can be developed quickly on top of this platform.

V.   EXPOSURE AND POSITION TRACKING APPLICATION

A major concern of financial regulators is that traders may put brokers at risk by exposing themselves to large, unbalanced

positions. While it is normal that traders maintain both long and short positions in the market, it is critical that the net position remains within a bound so that there is a margin of safety in the holdings. The ability to monitor positions and exposures in real-time can help avert finanical disaster.

This section describes how Algo-Logic's platform was used to track the real-time exposure and positions of multiple trading clients by parsing FIX execution reports. Figure 9 illustrates the system set-up of this application. AlgoLogic's system acts as a "bump in the wire" between broker and exchange, processing the FIX execution reports coming from the exchange while passing on orders coming from the involved brokers. A third 10GE interface connects the appliance with the control and logging host. This interface is used to read in normalized market data with price updates, to control the appliance, retrieve data, monitor status and log the actions performed by the card.
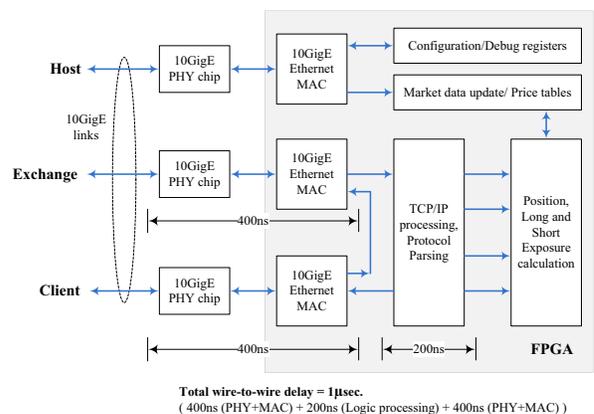

Figure 9   Exposure and Position-tracking Application built on Algo-Logic's HFT Platform

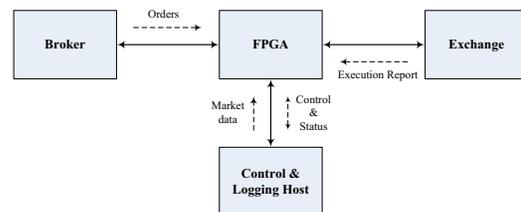### A. Exposure/Position Processing Circuit

Figure 10 Block Diagram of Exposure and Position Tracking System

All processing for the exposure and position tracking system is implemented entirely on the FPGA. The various components of the processing system are illustrated in Figure 10. All components are part of the low latency library.

The design includes three instances of the 10GE MAC to interface to the host, the broker and the exchange, respectively. The host interacts via the configuration/debug register module and can access the market data update/price table module. Current market prices for each security are received via UDP

from the exchange and stored within on-chip memory. These prices are needed to calculate exposure. Traffic from brokers and FIX execution reports from the exchange are passed from the MACs to the TCP/IP processing module. Along with network functions, this module also handles the processing of the financial protocols. For this application, the Financial Information eXchange (FIX) protocol is used. The parser extracts all pertinent FIX fields ("tags") from the execution report, including Security, Shares Filled, Price and Order Type. These correspond to FIX tags 55, 38, 44 and 54, respectively.

The Exposure and Position calculation module receives the traffic from the TCP/IP module and updates the market data/price tables using the procedure described in the following subsection. Finally, the FIX execution reports are forwarded to the clients providing details of order execution.

B. Position and Exposure Calculations

The exposure and positions are calculated in the FPGA from the extracted data according to the following formulas:

position = total number of shares (Tag 38) filled by the exchange

If it is a Buy order (Tag 54), then the position is categorized as "long". If it is a Sell/Short Sell order, then the position is "short". The net position per security (FIX Tag 55) is:

net position = (total long positions) – (total short positions)

Similarly, long exposure per security is defined as follows, where the current market price comes from the price table:

*long exposure (security) = (total long positions) * current market price*

*short exposure (security) = (total short positions) * current market price*

Exposure (security) across all sessions is defined as the product of net position and current market price. Net and gross market values can be calculated as follows:

*Net market value = sum[position/security across all sessions * price]*

*Gross market value = sum[abs(positions) * price]*

Once the Tags are extracted from the Execution reports and prices are read from the price tables, the exposure is calculated and position is updated based on the type of order.

C. User Interface

Position and exposure data are periodically forwarded to the control and logging host in UDP packets for visualization on a Graphical User Interface (GUI). The speed at which the screen is updated is limited only by the host's ability to capture logs and process them. The host updates its database and populates positions and exposures on a web-based GUI as shown in Figure 11. Exposure is plotted in the time domain.

The table on the left shows positions per security and dollar values based on current market price per security. Green indicates long positions and red indicates short positions. The table on the top-right visualizes long exposure per session, short exposure per session, net market value per session and gross market value per session. Dollar values are calculated based on latest prices received from the market. Finally, the graph in the bottom-right corner illustrates all exposures. The green and red lines represent the sum of long and short

exposures, respectively, across all sessions. Looking at this plot, a broker or regulator can instantly view total exposure.

Figure 11   Web-based GUI showing Exposure and Position

D. Results

The total delay from wire to receiving FIX payload from socket, parsing FIX execution reports and calculating these values can be significantly reduced by performing all these operations above within the FPGA. Rather than reading data across a memory bus, data is stored in fast on-chip memory and QDR-II directly attached to the FPGA. This memory is accessed without using the PCIe bus when an order (from client to exchange) enters FPGA and the table lookups and computation are performed directly in logic.

The total wire-to-wire latency in this application is **1μsec** with liertally no jitter. The breakup of 1 μsec total delay is 400ns (PHY+MAC, high-speed serial receiver) + 200ns (parsing/calculations) + 400ns (PHY+MAC, high-speed serial transmitter). With future generations of FPGAs these delays will further decrease. It is worth noting that the 1μs round-trip latency through this design is between one and two orders of magnitude lower than a recent software implementation [21].

As for throughput, all the processing on the FPGA is performed at the full 10Gbps line rate. When using FIX, the average size of a message is 150 bytes (the exact value depends of the value of the fields). OUCH messages are smaller, and range in size between 12 to 82 bytes in length, whereby multiple messages can appear in a TCP/IP flow. When a single FIX message appears in a TCP packet, the total size of the packet is 204 bytes. The throughput is therefore roughly 6.1M FIX messages/second.

In regards to development, it is important to note that the application could be completely assembled out of existing library components. The Low-Latency Library components were simply connected together to form the infrastructure, including FIX parsing. Secondly, the integration of the hardware was completed within a month. Most importantly, the Low-Latency Library and hardware platform were pre-verified, greatly simplifying the task of verifying the final design. Overall, the IP library reduced development time considerably.

VI.   CONCLUSIONS

In the race to minimize latency in high frequency trading, traders, brokers and exchanges are exploring a range of technologies to build platforms, monitor and manage risk and improve the efficiency of trading.

Software approaches that utilize CPUs with high performance network interface cards provide part of the solution. Through optimized bus logic and device driver software, they reduce the time required to transfer a message from the network to the CPU then back to the network in just a few microseconds. Smart NICs go one step further by offloading some of the processing to a Field Programmable Gate Array (FPGA) on the network adapter.

The advantage of developing financial applications in software is a typically short development time. A challenge with software is that it is hard to achieve low end-to-end latency and jitter. A custom-designed datapath in an FPGA circuit offers the benefits of minimal, deterministic processing times down to clock cycles. However, the benefits of FPGAs have traditionally come at the cost of a longer development time. The contribution of this paper is to provide a pre-built gateware library that allows financial applications to be built on standard FPGA cards (such as the NetFPGA-10G) without the burden of an extended development period.

Algo-Logic's Low-Latency Library described in this paper includes infrastructure components and domain-specific gateware that extract data from market data streams, including FIX, OUCH, XPRS, ArcaDirect, LSE and BATS BOE. For order processing systems that also need to make decisions based on price, Algo-Logic's gateware includes a module that receives normalized market data via UDP and stores current prices using on-chip memory so that logic can use these values during order processing.

To demonstrate the utility of this gateware library, a complete application has been developed which tracks the exposure and positions of multiple traders as orders are transferred over and back to the stock exchange. Incoming FIX messages are decoded, analyzed and forwarded onward in real time with a stable end-to-end latency of 1µs. This is up to two orders of magnitude lower than software approaches achieve.

REFERENCES

[1] P. Gomber, B. Arndt, M. Lutat and T. Uhle, "High Frequency Trading" (report commissioned by Deutsche Börse Group), Goethe Universitat Frankfurt am Main, March 2011 http://www.frankfurt-main-finance.de/de/finanzplatz/daten-studien/studien/High-Frequency-Trading.pdf

[2] J.W. Lockwood, N. McKeown, G. Watson, G. Gibb, P. Hartke, J. Naous, R. Raghuraman and Jianying Luo, "NetFPGA - An Open Platform for Gigabit-rate Network Switching and Routing", *Microelectronic Systems Education, 2007. MSE '07. IEEE International Conference on*, pp. 160-161, 3-4 June 2007

[3] G. Gibb, J.W. Lockwood, J. Naous, P. Hartke and N. McKeown, "NetFPGA—An Open Platform for Teaching How to Build Gigabit-Rate Network Switches and Routers," *Education, IEEE Transactions on* , vol.51, no.3, pp.364-369, Aug. 2008

[4] A. Gupte and J.W. Lockwood, "Precise Precise Latency Comparison Module for the NetFPGA", workshop at the 2010 North American NetFPGA Developers' Workshop, http://netfpga.org/tutorials/WorkshopNorthAmerican2010/pdf/NetFPGA_Dev_2010_Precise_Latency_Comparison_Module.pdf

[5] Cypress Quad Data Rate (QDR-II) Static Random Access Memory (SRAM), http://www.cypress.com/?id=107

[6] Myricom DBL (Datagram Bypass Layer) Documentation, http://myri.org/dbl.html

[7] Datasheet for the QLogic 7300-series Network Adapters, http://www.qlogic.com/Resources/Documents/DataSheets/Adapters/DataSheet_QLE7300Series.pdf

[8] M. Blott, J. Ellithorpe, N. McKeown, K. Vissers and H. Zeng, "FPGA Research Design Platform Fuels Network Advances", *Xilinx Xcell Journal*, Issue 73, September 2010

[9] N. Lobo, V. Malik, C. Donnally, S. Jahne and H. Jhaveri, "Evaluating the Latency Impact of IPv6 on a High Frequency Trading System", University of Colorado, May 2012

[10] M. Sadoghi, M. Labrecque, H. Singh, W. Shum and H.A. Jacobsen, "Efficient Event Processing through Reconfigurable Hardware for Algorithmic Trading," *Journal Proceedings of the VLDB Endowment,* vol. 3,no. 1-2, pp. 1525-1528, September 2010

[11] G.W. Morris, D.B. Thomas and W. Luk, "FPGA Accelerated Low-Latency Market Data Feed Processing", *High Performance Interconnects, 2009 (HOTI 2009). 17th IEEE Symposium on,* pp. 83-89, 25-27 Aug. 2009

[12] Report by the Aite Group, "A New World Order: The High Frequency Trading Community and Its Impact On Market Structure", 2009

[13] R. Martin, "Wall Street's Quest to Process Data at the Speed of Light", Information Week, 21st April 2007

[14] O*U*C*H 4.2 Protocol Specification, NASDAQ Inc., http://www.nasdaqtrader.com/content/technicalsupport/specifications/TradingProducts/OUCH4.2.pdf

[15] XPRS 1.24 Protocol Specification, Direct Edge, http://www.directedge.com/Portals/0/docs/Connect/Direct%20Edge%20XPRS%20API%20Manual%20V%201.24.pdf

[16] ARCADirect 4.0 Protocol Specification, NYSE Arca, http://www.nyse.com/pdfs/ArcaDirectSpecVersion4_0.pdf

[17] Native Trading Gateway 10.1 Protocol Specification, London Stock Exchange (LSE), http://www.londonstockexchange.com/products-and-services/millennium-exchange/millennium-exchange-migration/mit203v101.pdf

[18] BATS US Options BOE Specification 1.5.2, BATS Global Markets Inc., http://www.batsoptions.com/resources/membership/BATS_US_Options_BOE_Specification.pdf

[19] R. Iati, "The Real Story of Trading Software Espionage", TABB Group Perspective, July 10th, 2009

[20] AMBA AXI4-Stream Protocol Specification, ARM Ltd., http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ihi0051a/index.html

[21] OnX Enterprise Solutions, "Report: High Performance Trading – FIX Messaging; Testing for Low Latency", Jan 2012, http://fixglobal.com/content/high-performance-trading-fix-messaging-testing-low-latency

[22] R. Mueller, J. Teubner and G. Alonso, "Streams on wires: a query compiler for FPGAs", Proc. VLDB Endowment, vol. 2, no. 1, Aug 2009, pp. 229 – 240

[23] M. Kearns, A. Kulesza and Y. Nevmyvaka, "Empirical Limitations on High Frequency Trading Profitability", Working Paper Series, Social Science Research Network (SSRN), September 17, 2010

[24] H. Subramoni, F. Petrini, V. Agarwal, D. Pasetto, "Streaming, Low-latency Communication in On-line Trading Systems", *IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010*, pp.1-8, 19-23 April 2010

[25] C. Starke, V. Grossman, L. Wienbrandt, M. Schimmler, "An FPGA Implementation of an Investment Strategy Processor", Procedia Computer Science, Volume 9, 2012, Pages 1880-1889

[26] R. Pottahuparambil, J. Coyne, J. Allred, W. Lynch and V. Natoli, "Low-latency FPGA Based Financial Data Feed Handler", *2011 IEEE 19th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pp. 93-96, 1-3 May 2011