Hardware Accelerated FIX Order Cancel System

Terry Stratoudakis

Wall Street FPGA, LLC

www.WallStreetFPGA.com

New York City

March 2011

Abstract

This paper presents a Hardware Accelerated FIX Order Cancel System. The open source FIX Engine, QuickFIX[1] is accelerated using Field Programmable Gate Array[2] (FPGA) technology. The acceleration is performed by an FPGA based network card which is optimized for QuickFIX. FIX 4.2 Order Cancel[3] messages are generated entirely inside the FPGA. The latency from the Order Cancel trigger to when the first byte is on the wire is 314 nanoseconds. The latency from the Order Cancel trigger to when the first FIX Order Cancel message is entirely on the wire is 1,874 nanoseconds.

Introduction

The majority of trading is increasingly done electronically using computers; see Figure 1. Peak trading periods provide among the best trading opportunities for profits but also present the largest risk for potential losses. During peak trading periods, market data that trading systems must consume and process surges to the point where trading systems slow down and become ineffective.

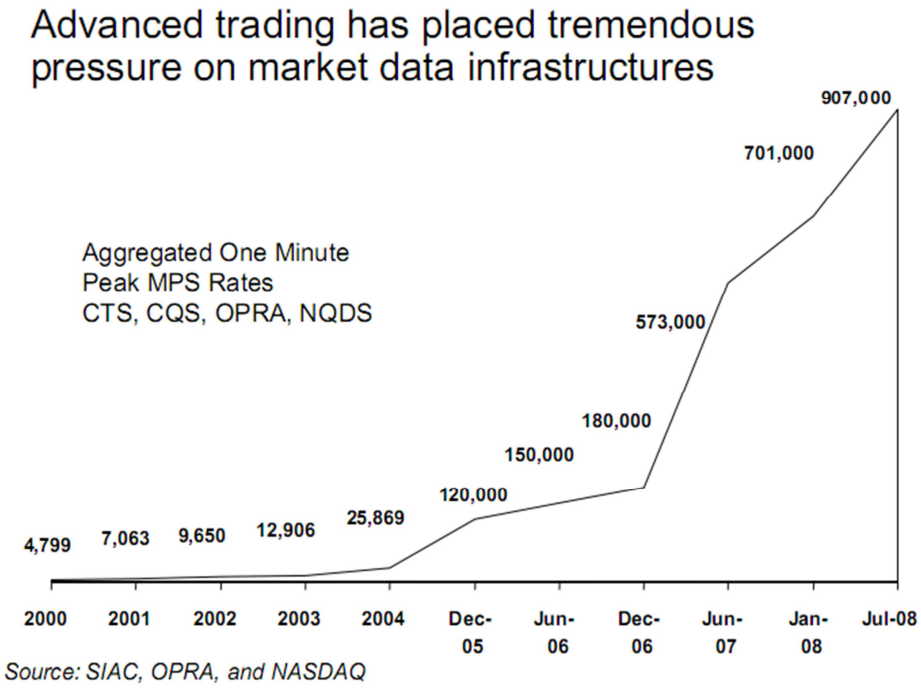

Figure 1 – Aggregate One Minute Peak Messages Per Second

[1] QuickFIX Engine, http://www.quickfixengine.org/
[2] Wikipedia, *Field Programmable Gate Arrays*, http://en.wikipedia.org/wiki/Field-programmable_gate_array
[3] FIX Protocol Limited (FPL), FIX 4.2 Order Cancel Message Format, http://www.fixprotocol.org/FIXimate3.0/en/FIX.4.2/body_495470.html

High Frequency Trading (HFT) is impacting market dynamics and generated interesting debates.[4] For some, HFT is a relative term. What is called HFT today may be the common form of trading in the future. And whether or not a firm engages in HFT, they will certainly need to protect their assets from events such as the "Flash Crash" of May 6, 2010. [5] It is possible that a trading firm using an FPGA based Order Cancel system could have exited the market faster than any other trading firm – thereby reducing losses on days such as this one; see Figure 2 below.

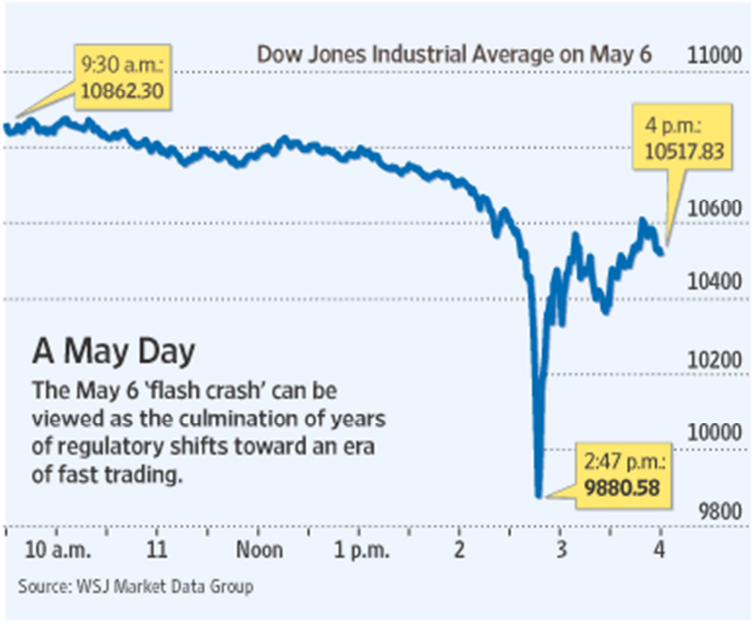

Figure 2 – Dow Jones Industrial Average on May 6, 2010 ("Flash Crash")

FIX Protocol

Financial firms communicate market and trade data via messaging standards such as the FIX protocol.[6] The Financial Information eXchange (FIX) Protocol is a messaging standard for the real-time electronic exchange of securities transactions. The FIX standard is managed by FIX Protocol Limited (FPL), an international non-profit standards body with members from all aspects of the financial services industry.

[4] Finextra Research, *HFT makes up 77% of UK market – Tabb* http://www.finextra.com/news/fullstory.aspx?newsitemid=22194 (24 January, 2011)
[5] CFTC and SEC, *Findings Regarding the Market Events of May 6, 2010 - Report of the Staffs of the CFTC and SEC to the Joint Advisory Committee on Emerging Regulatory Issues,* http://www.sec.gov/news/studies/2010/marketevents-report.pdf (September 30, 2010).
[6] FIX Protocol Limited, *What is FIX?* http://www.fixprotocol.org/what-is-fix.shtml

Software known as FIX Engines are used to process and generate FIX messages. QuickFIX is the de facto open source FIX Engine. QuickFIX as well as commercial "closed source" FIX Engines are used by financial firms.

Financial firms are turning to High Performance Computing (HPC) technology to provide that extra advantage over their competitors. Every level of software is optimized, and in recent years, firms are optimizing the hardware of their trading systems through the use of reconfigurable hardware.

FPGAs: Reconfigurable Hardware

Reconfigurable hardware such as Field Programmable Gate Array (FPGA) technology is used to optimize trading systems at the network level. FPGAs can aid in the generation and processing of network data thereby offloading certain tasks from the software of a system. The most common usage of FPGAs in finance is Market Data Handling.[7] FPGAs can have up to 1000 cores[8] for processing data in parallel and do not have the jitter introduced by operating systems and instruction fetching.[9]

An FPGA is programmed using a Hardware Description Language (HDL)[10] such as Verilog or VHDL. Not all algorithms can be implemented on an FPGA. This is partly due to the nature of FPGAs as a technology and also due to the low level aspect of HDL. HDL requires more knowledge of the target hardware than traditional programming such as C or C++. HDL coding can result in development times of 3 to 5 times more if the algorithm can even be implemented on an FPGA. These languages are difficult to learn and result in very lengthy source code files that often accomplish very little with a lot of effort. For example, the VHDL code for calculating the square root of a number can take anywhere from 117 lines[11] to 396 lines[12] of code.

The FIX Protocol is string based, lending itself to large benefits from an FPGA. String functions are among the least efficient in a CPU.

---

[7] Pete Harris, A-Team Group, *FPGAs, Established for Market Data, Now Being Leveraged for Transactions, Pre-Trade Risk*, http://www.a-teamgroup.com/article/fpgas-established-for-market-data-now-being-leveraged-for-transactions-pre-trade-risk/ (December 7, 2010)

[8] Kit Eaton, *1,000 Core CPU Achieved: Your Future Desktop Will Be a Supercomputer*, http://www.fastcompany.com/1714174/1000-core-cpu-achieved-your-future-desktop-will-be-a-supercomputer (Tuesday Jan 4, 2011).

[9] Scott Sirowy and Alessandro Forin, Microsoft Research, *Where's the Beef? Why FPGAs Are So Fast - Technical Report - MSR-TR-2008-13*, http://research.microsoft.com/pubs/70636/tr-2008-130.pdf (September 2008)

[10] http://en.wikipedia.org/wiki/Hardware_description_language

[11] VHDL Reference Material: Simple parallel 8-bit sqrt using one component; University of Maryland, Baltimore County, Department of Computer Science & Electrical Engineering; http://www.cs.umbc.edu/portal/help/VHDL/samples/samples.shtml#sqrt8; Last updated: August 20, 2007.

[12] VHDL Reference Material: 32-bit parallel integer square root; University of Maryland, Baltimore County, Department of Computer Science & Electrical Engineering; http://www.cs.umbc.edu/portal/help/VHDL/samples/samples.shtml#sqrt32; Last updated: August 20, 2007.

Technology Platform

The technologies used in this system are PXI, FlexRIO, and LabVIEW FPGA.

PXI

PXI (PCI eXtensions for Instrumentation) is an open and standardized computer bus using the CompactPCI form factor.[13] PXI is based on the PCI bus with integrated timing and synchronization that is used to route clocks and triggers internally. PXI was developed in 1997 and launched in 1998. Today, PXI is governed by the PXI Systems Alliance (PXISA), a group of more than 70 companies chartered to promote the PXI standard, ensure interoperability, and maintain the PXI specification. PXISA is also responsible for the PXI Express bus which is based on the PCI Express computer bus found in newer computers.

PXI systems are composed of three basic components -- chassis, controller, and peripheral modules. For example, an 18 slot 19 inch rack-mountable chassis holds one controller and 17 single slot modules. The controller contains laptop sized components for compactness and can run Windows, Linux, and various real-time operating systems. There are 3U and 6U modules.

Since PXI is based on standard PC technologies such as Windows and the PCI bus, integrating a PXI system to these systems is similar to integrating these systems with a PC. PXI also supports a broad range of Compact PCI products as they are both based on the same form factor.

The PXI bus combines the high-speed PCI bus with timing and synchronization. The PXI trigger bus consists of 8 shared trigger bus lines, a low-skew star trigger, and a common 10 MHz system reference clock. Using these synchronization features, one can pass trigger, clock, and other signals between PXI modules to make the accurate, high-performance measurements.

FlexRIO

FlexRIO is a PXI and PXI Express based reconfigurable hardware platform developed by National Instruments.[14] It features two parts: FlexRIO field-programmable gate array (FPGA) modules and FlexRIO adapter modules. Together, they form a high-performance, reconfigurable hardware system programmable with LabVIEW FPGA software and without Hardware Description Language (HDL) design knowledge.[15]

NI FlexRIO FPGA Modules

FlexRIO FPGA modules utilize a Virtex-5 FPGA with up to 512 MB of onboard DDR2 DRAM. FlexRIO FPGA modules come in PXI and PXI Express formats and interface to FlexRIO adapter modules that provide I/O to the FPGA. PXI FlexRIO modules have three (3) DMA channels[16] for high-speed data streaming while the PXI Express FlexRIO modules have sixteen (16) DMA channels.[17] See Figure 3.

The adapter module interface consists of 132 lines of general-purpose digital I/O directly connected to FPGA pins, in addition to the power, clocking, and supplementary circuitry necessary to define the interface. These

[13] PXI Systems Alliance (PXISA) http://www.pxisa.org/
[14] http://www.ni.com/flexrio/
[15] http://zone.ni.com/devzone/cda/tut/p/id/7962#toc7
[16] http://sine.ni.com/nips/cds/view/p/lang/en/nid/206640
[17] http://sine.ni.com/nips/cds/view/p/lang/en/nid/208164

132 lines can be configured for single-ended operation at rates of up to 400 Mbits/s and differential operation at rates of up to 1 Gbit/s for a maximum I/O bandwidth of 66 Gbits/s (8.25 GB/s). All lines are routed with controlled-impedance, matched-length traces, and the differential pairs are routed together.

Peer-to-Peer Data Streaming is unique to PXI Express NI FlexRIO FPGA modules. This allows banks of up to four (4) PXI Express NI FlexRIO FPGA cards to deterministically communicate with each other. They are capable of streaming data between modules at rates above 800 MB/s and latencies of no more than 10 microseconds as data is not routed through the host chipset. Up to 16 such streams are supported, simplifying complex multi-FPGA communication schemes without taxing host CPU resources.[18]

FlexRIO cards have RTSI, or Real Time Signal Integration, which enables cards to be synchronized using the PXI bus. RTSI enabled devices can communicate directly over a low latency electrical connection.

NI FlexRIO FPGA modules are accessible by Windows, Linux, PharLap, and VxWorks Operating Systems via the NI RIO drivers; version 3.5.1 of the RIO drivers were used for this application.

Figure 3 – FlexRIO FPGA Module

FlexRIO Adapter Modules

FlexRIO Adapter modules from National Instruments and third parties interface with FlexRIO FPGA modules through a card-edge connector that routes the necessary FPGA signals to the adapter module. Custom adapter modules can be developed with the NI FlexRIO Adapter Module Development Kit (MDK).[19]

The FlexRIO Adapter Module Development Kit (MDK) can be used to build custom I/O to meet exact application needs. The FlexRIO card edge connector offers direct access to the raw digital I/O pins of the FPGA. Each pin is capable of low-voltage differential signaling (LVDS) rates up to 1 Gb/s and single-ended

[18] http://zone.ni.com/devzone/cda/tut/p/id/10801
[19] http://sine.ni.com/nips/cds/view/p/lang/en/nid/206645

rates up to 400 Mb/s. The adapter modules are interchangeable and define the I/O available in the LabVIEW FPGA programming environment.

In this paper the Prevas Mimas Gigabit Ethernet Adapter is used, see Figure 4 below.[20]

Figure 4 – Prevas Mimas connected to NI FlexRIO Module

LabVIEW FPGA

FPGAs are field programmable which saves on development and modification costs. Prior to FPGAs, custom logic required schematic design which led to Register Level Transfer (RTL). RTL which was replaced by Verilog and VHDL. Increased FPGA capacity requires a higher level of abstraction. In the past decade, industry has developed High Level HDLs. This allows for more complex algorithms to be implemented in a more timely fashion.

National Instruments' LabVIEW FPGA platform provides a graphical approach to developing logic for an FPGA. Complex financial algorithms can be programmed onto FPGAs without in-depth knowledge of digital design or complex Electronic Design Automation (EDA) tools. LabVIEW is distinctly suited for FPGA programming because it provides an intuitive depiction of the inherent parallelism that FPGAs provide. Using a high level, graphical development environment (see Figure 5) of LabVIEW FPGA reduces development time without compromising the performance gains of using an FPGA.[21]

[20] http://www.prevas.com/ethernet_simulator.html
[21] http://www.ni.com/fpga/

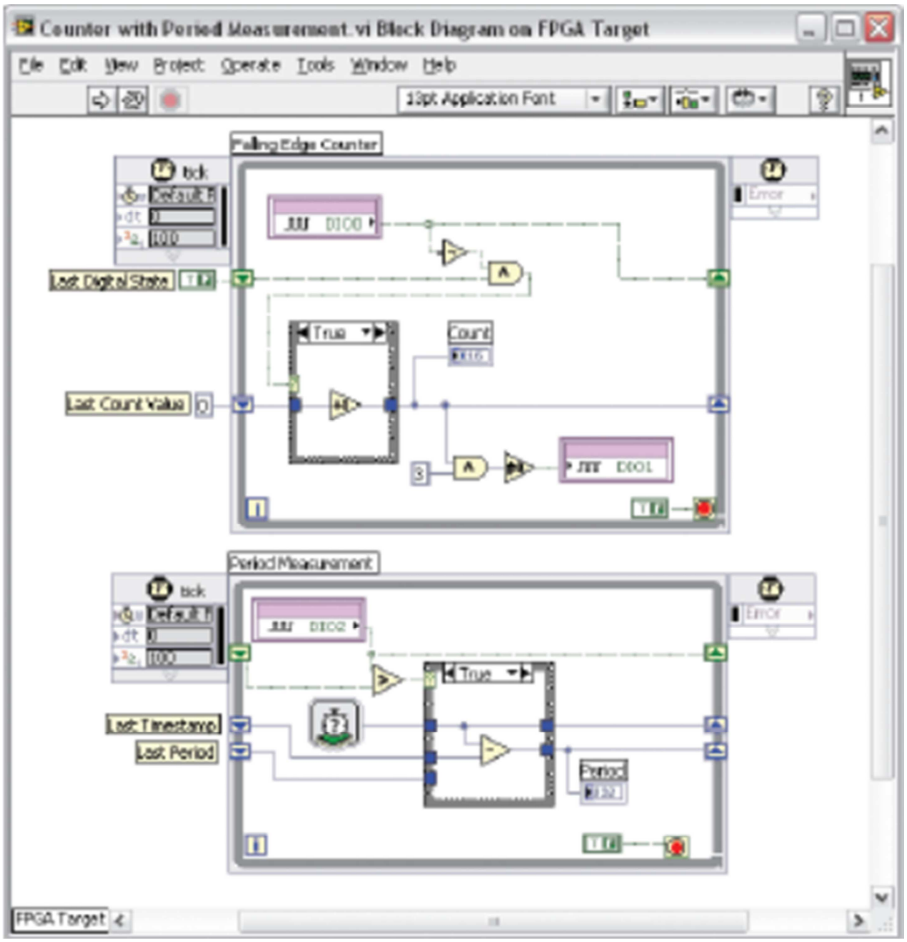

Figure 5 – LabVIEW FPGA sample Block Diagram

Under the hood, the LabVIEW FPGA module uses code generation techniques to synthesize the graphical development environment to FPGA hardware which ultimately runs the FPGA synthesis tools. The single-cycle timed loops (SCTL)[22] in LabVIEW FPGA provide a level of determinism guaranteed to execute within a specified time period of at least 40 MHz. In this paper, a SCTL running at 125 MHz was used.

Custom hardware can be used to create unique timing and triggering routines, ultrahigh-speed control, interfacing to digital protocols and applications requiring high-speed hardware reliability and tight determinism. In this paper, LabVIEW FPGA is used to create a *protocol aware hardware based system*.[23]

[22] http://zone.ni.com/devzone/cda/tut/p/id/3749#toc5
[23] http://zone.ni.com/devzone/cda/tut/p/id/9693#toc2

Scenario

A broker-dealer is connected to an exchange. The trade messages between the broker-dealer and the exchange use the FIX protocol format. The broker-dealer submits orders which the exchange seeks to match. All orders from the broker-dealer are open until a matching order enters the exchange. Matched orders are sent back to the broker-dealer as executed. At some point, the broker-dealer detects an "event" resulting in its need to cancel all open orders. It is assumed that the detected event is one which will cause many other broker-dealers to wish to cancel their open orders at the same time, so time would be of the essence and those who cancel first will reduce their potential losses. See Figure 6 below.

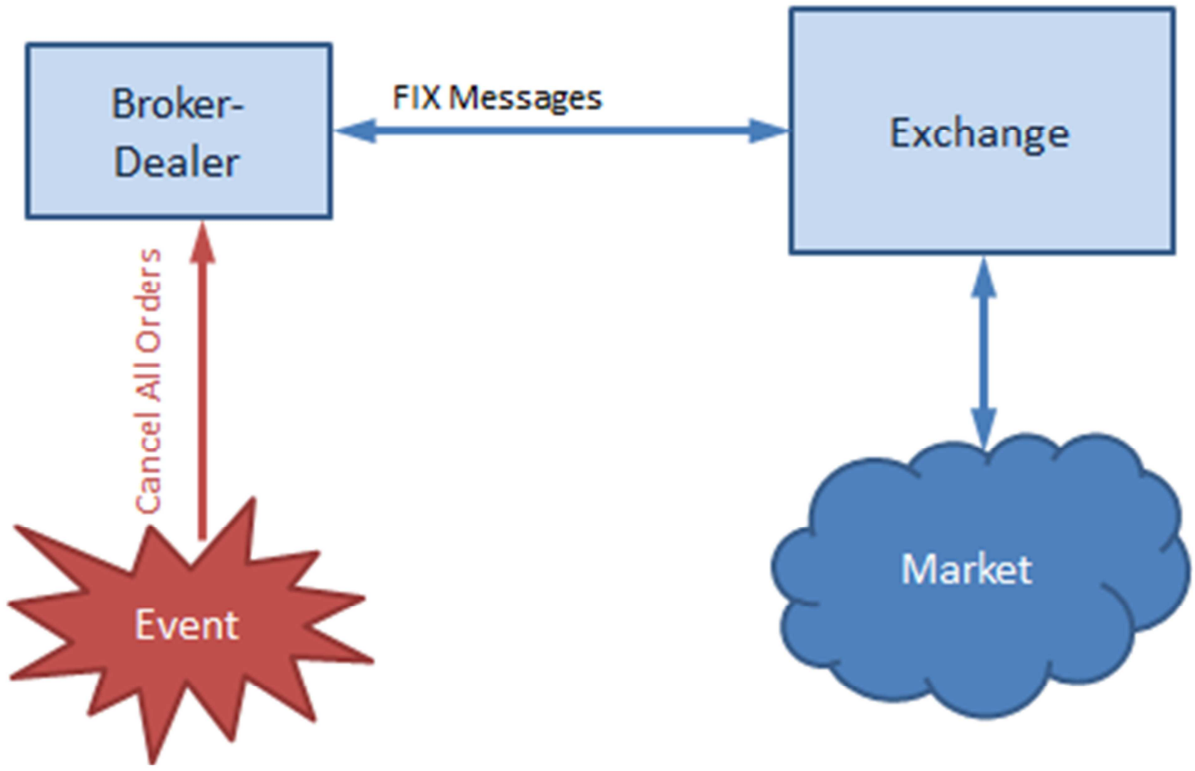

Figure 6: Broker-Dealer maintaining a connection to an Exchange by using FIX messages. Exchange is also connected to other Markets and accepts Orders from there as well. Upon a special trigger "Event", Broker-Dealer sends a series of Cancel Orders to the Exchange to get it out of the Market as quickly as possible.

Implementation

This was implemented using two computers connected directly via a cross over Ethernet cable. They communicate using FIX 4.2 running over TCP/IP at 1 Gigabit/second. Both computers are running Microsoft Windows XP on Intel x86 CPUs. One computer represents the Broker-Dealer, and the other the Exchange. The Broker-Dealer sends to the Exchange several buy or sell orders such that they do not execute. Upon detection of a trigger, the Broker-Dealer sends FIX Order Cancel messages for all open orders. *See Figure 7.*

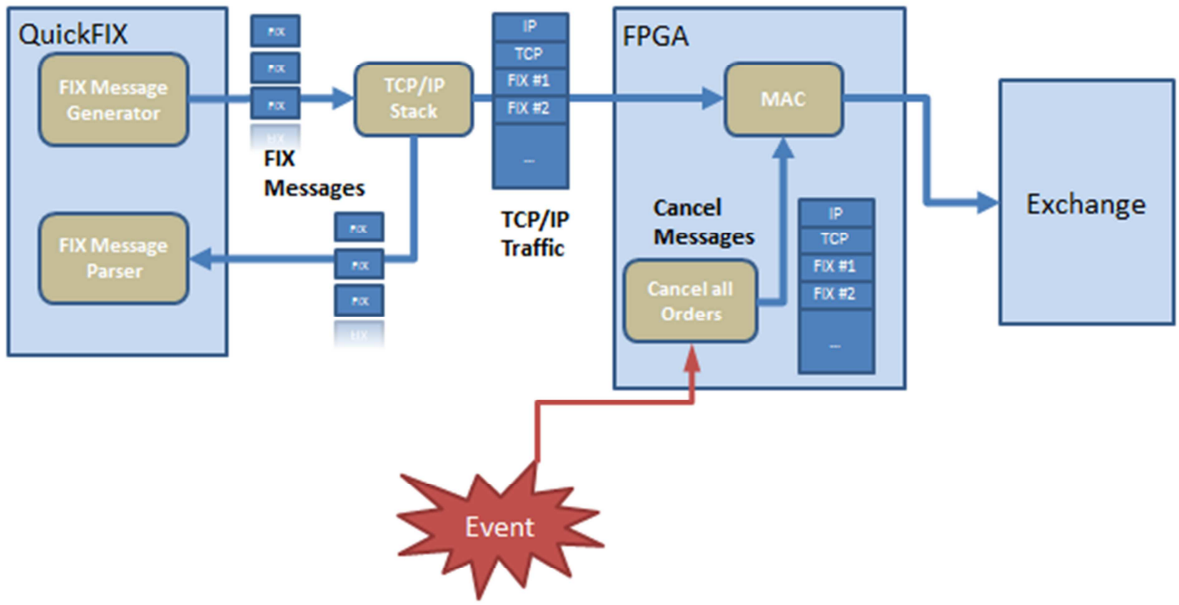

Figure 7: QuickFIX in normal operation with FPGA-based Network Interface Card. Event triggers directly inside FPGA, which generates FIX Cancel Requests and places them inside valid TCP/IP packets of existing FIX session maintained by QuickFIX FIX Engine. FPGA MAC then transfers packets to PHY for transmission to Exchange

Broker-Dealer Computer

The Broker-Dealer computer is comprised of a computer connected to a NI PXI-1033 5-Slot PXI Chassis with Integrated MXI-Express Controller. Two PXI cards are plugged into the PXI chassis; a FlexRIO PXI-7953 FPGA card with a Xilinx Virtex-5 LX85 FPGA and a NI PXI-6070E DAQ card. The FlexRIO card has a Prevas Mimas Gigabit Ethernet Adapter which has two RJ-45 connectors. The FlexRIO and Prevas Mimas adapter combine to function as the Network Interface Card (NIC) of the Broker-Dealer computer. All network traffic goes through the FlexRIO card and one of the ports of the Prevas Mimas Dual Gigabit adapter (the other port is unused for this application).

The Prevas Mimas adapter has a 'PHY' chip which converts inbound electrical signals on an RJ-45 cable to Ethernet Frames and vice versa. The Ethernet frames come into the Xilinx Virtex-5 LX85 FPGA in the form of bytes (U8).

Under normal conditions, the FlexRIO FPGA card allows the layer 2 Ethernet frame data to go out via the Prevas Mimas adapter and in by passing it into the PXI bus so that it is accessible by programs that interface to the NI RIO drivers.

QuickFIX normally interfaces with Winsock functions which interface with Windows' closed source TCP/IP stack. Being closed source, it cannot interface to the RIO drivers and therefore lwip, an open source TCP/IP stack, was selected instead. All of QuickFIX's calls to Winsock were modified to call the lwip TCP/IP stack which in turn was modified to interface with the RIO drivers.

The NI PXI-6070E Data Acquisition card was used to receive and transfer the Cancel Orders trigger. One of the PXI RTSI lines was used to make a direct electrical connection between this and the FlexRIO FPGA card. An external button was connected to one of the DAQ card's digital inputs such that when pressed, a digital signal would be read by the FPGA. This is read purely in hardware with no interaction by the computer or the software.

Exchange Computer

The Exchange computer is comprised of another computer running an unmodified version of QuickFIX running in server mode. Network interfacing is done via the native Gigabit Ethernet port. Its function is to hold an order open until either a matching one arrives, or the order is cancelled.

Results

Several non-matching orders are entered into the Broker-Dealer's QuickFIX program. It sends them to the Exchange computer which holds them as open orders. The Broker-Dealer's QuickFIX program maintains a list of its own open orders. Whenever any aspect of the open orders changes on the Broker-Dealer, the FPGA receives a copy of the cancellation information for each open order. Below is a sample FIX 4.2 ORDER SINGLE (tag 35=D) message in offset hex and then the readable characters only.

```
0000  00 04 4B 02 F6 23 00 1E 65 D6 A4 D0 08 00 45 00
0010  00 CF 4E C6 40 00 80 06 94 76 0A 00 01 6D 0A 00
0020  01 80 C3 07 13 8A 7F 60 31 DD F3 05 92 F4 50 18
0030  44 12 22 74 00 00 38 3D 46 49 58 2E 34 2E 32 01
0040  39 3D 31 34 34 01 33 35 3D 44 01 33 34 3D 32 01
0050  34 39 3D 43 4C 49 45 4E 54 31 01 35 32 3D 32 30
0060  31 30 30 36 30 34 2D 32 33 3A 35 38 3A 34 38 2E
0070  35 35 36 01 35 36 3D 4F 52 44 45 52 4D 41 54 43
0080  48 01 31 31 3D 31 32 37 35 36 39 35 39 32 38 35
0090  31 31 01 32 31 3D 31 01 33 38 3D 39 38 01 34 30
00A0  3D 32 01 34 34 3D 31 30 32 2E 37 01 35 34 3D 31
00B0  01 35 35 3D 49 42 4D 01 35 39 3D 30 01 36 30 3D
00C0  32 30 31 30 30 36 30 34 2D 32 33 3A 35 38 3A 34
00D0  38 2E 35 35 31 01 31 30 3D 31 38 30 01
```

8=FIX.4.2	56=ORDERMATCH	54=1
9=144	11=1275695928511	55=IBM
35=D	21=1	59=0
34=2	38=98	60=20100604-23:58:48.551
49=CLIENT1	40=2	10=180
52=20100604-23:58:48.556	44=102.7	

Below is a sample FIX 4.2 ORDER CANCEL REQUEST (tag 35=F) message in offset hex and then the readable characters only.

```
0000  00 04 4B 02 F6 23 00 1E 65 D6 A4 D0 08 00 45 00
0010  00 C8 4E C8 40 00 80 06 94 7B 0A 00 01 6D 0A 00
0020  01 80 C3 07 13 8A 7F 60 32 84 F3 05 93 A0 50 18
0030  43 66 D2 0B 00 00 38 3D 46 49 58 2E 34 2E 32 01
0040  39 3D 31 33 37 01 33 35 3D 46 01 33 34 3D 33 01
0050  34 39 3D 43 4C 49 45 4E 54 31 01 35 32 3D 32 30
0060  31 30 30 36 30 34 2D 32 33 3A 35 38 3A 35 36 2E
0070  31 31 32 01 35 36 3D 4F 52 44 45 52 4D 41 54 43
0080  48 01 31 31 3D 31 32 37 35 36 39 35 39 33 36 31
0090  31 30 01 33 38 3D 39 38 01 34 31 3D 31 32 37 35
00A0  36 39 35 39 32 38 35 31 31 01 35 34 3D 31 01 35
00B0  35 3D 49 42 4D 01 36 30 3D 32 30 31 30 30 36 30
00C0  34 2D 32 33 3A 35 38 3A 35 36 2E 31 31 32 01 31
00D0  30 3D 32 30 36 01
```

8=FIX.4.2	56=ORDERMATCH	60=20100604-23:58:56.112
9=137	11=1275695936110	10=206
35=F	38=98	
34=3	41=1275695928511	
49=CLIENT1	54=1	
52=20100604-23:58:56.112	55=IBM	

Pressing the button connected to the PXI-6070E DAQ card generates a Cancel Orders trigger, which causes the FPGA to generate one FIX Order Cancel message for each of the open orders. The FIX message(s) are then injected into the live TCP session that already exists between the Broker-Dealer and Exchange machines. The Exchange computer receives the FIX Order Cancel messages, not realizing that those messages were created by the FPGA [and not by QuickFIX on the Broker-Dealer computer] it cancels all orders referenced as if the Broker-Dealer instructed it to do so.

For all normal FIX traffic, the QuickFIX application handles the creation of each FIX message, while the software TCP/IP stack handles the creation of TCP segments and IP packets. In the situation where a Trigger event has just occurred, the FPGA handles all the tasks of QuickFIX and the TCP/IP software by generating the FIX message(s), TCP segment(s), and IP packet(s). As the final payload is being transferred to the PHY, the Ethernet frame and CRC is also calculated by the FPGA.

Conclusion

Field Programmable Gate Array (FPGA) technology has been established for Market Data[24] and is now finding applications such as trade message generation. Trade volume and data increases are outpacing trading system technology. Financial firms seek to optimize every aspect of the trading system. Until recently, software has been the focus of optimizations.

Hardware optimization provides the lure of creating "the world's fastest and most deterministic trading system" and is the last frontier of customization. This is not without cost as a completely customized hardware optimized system can easily become a *zero-sum* game.

Using off the shelf FPGA boards and a High Level Hardware Description Language (HDL) such as LabVIEW FPGA, one can develop hardware accelerated trading systems with managed risk and costs. PXI is an open platform that can integrate FPGA based network data processing/generation and IEEE-1588v2 (2008) timestamps from a GPS time source for latency measurements. PXI Express FPGA boards can efficiently communicate amongst themselves [without the host CPU] using Peer-to-Peer Data Streaming to provide multi-FPGA trading systems.

Financial firms can now optimize both software and hardware of their trading systems to provide further differentiation and increase their competitiveness.

[24] Pete Harris, A-Team Group, FPGAs, *Established for Market Data, Now Being Leveraged for Transactions, Pre-Trade Risk*, http://www.a-teamgroup.com/article/fpgas-established-for-market-data-now-being-leveraged-for-transactions-pre-trade-risk/ (December 7, 2010)

About Wall Street FPGA, LLC

Wall Street FPGA, LLC (www.WallStreetFPGA.com) is a New York City-based boutique financial technology firm that delivers hardware accelerated, low latency, high throughput financial trading and analytics solutions. Wall Street FPGA, LLC combines knowledge of capital markets and disruptive technology. For more information and to schedule a demonstration, contact Terry Stratoudakis: terry@WallStreetFPGA.com or +1 (347) 228-7379. Wall Street FPGA, LLC is a member of the FIX Protocol Limited.

The Author

Terry Stratoudakis

Terry Stratoudakis, P.E. has over twelve years experience in automation. He specializes in hardware acceleration using FPGAs for Monte Carlo methods, complex/real matrix math, and pattern matching for use in various industries. Current projects include hardware acceleration of trading systems, market data analysis, and order entry as well as low latency measurement systems.

Terry is Executive Director of Wall Street FPGA, LLC. Terry is the co-founder of ALE System Integration, a National Instruments Certified Partner. He worked at Underwriters Laboratories (UL) designing automated systems for product safety testing; the systems interfaced with enterprise systems as well as a wide range of equipment. He taught instrumentation as an Assistant Adjunct Professor at the New York City College of Technology.

Terry holds a Masters of Science and Bachelors of Science in Electrical Engineering from Polytechnic University located in Brooklyn, New York ("NYU-Poly"). He is a New York State licensed Professional Engineer and a National Instruments Certified LabVIEW Architect and Certified Professional Instructor. Terry is a member of the IEEE Long Island Consultants Network and Instrumentation & Measurement Society. He is a member of the Global Technical Committee (GTC), High Frequency Trading Working Group, and Inter-Party Latency Working Group of the FIX Protocol Limited (FPL).