

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

ACTIV FINANCIAL SYSTEMS, INC.,
Petitioner,

v.

IP RESERVOIR, LLC,
Patent Owner.

Case No. TDB
Patent No. 10,062,115

DECLARATION OF BERNARD S. DONEFER

ACTIV Exhibit 1004 ACTIV v IP Reservoir
--

TABLE OF CONTENTS

I.	PERSONAL AND PROFESSIONAL BACKGROUND	1
II.	MATERIALS REVIEWED AND CONSIDERED	6
III.	MY UNDERSTANDING OF PATENT LAW.....	8
	A. Claim Construction.....	8
	B. Prior Art & Priority	9
	C. Obviousness.....	10
IV.	OVERVIEW OF RELEVANT TECHNOLOGY	13
	A. Financial Markets and Financial Data.....	13
	1. Order Books	16
	2. Electronic Exchanges Electronic Data Feeds.....	21
	B. Coprocessor Technology	26
	1. Field Programmable Gate Arrays (FPGAs)	29
V.	SUMMARY OF THE '115 PATENT.....	31
	A. Level of Ordinary Skill in the Art	38
	B. Overview of the Challenged Claims	39
VI.	DISCLOSURE OF PARSONS	39
	A. Parsons Described the Same Reconfigurable FPGA's for Performing the Same Processing of Financial Market Data Prior to the '115 Patent	39
	B. Parsons's Caching Modules (OBC, LVC, GVC).....	48
	C. Parsons's Order Book Server	49
	1. The OBS FAM Pipeline	54
	D. Parsons's Example LVC Memory Manager and Message/Record Updater	57
VII.	CLAIMS 43 AND 44 ARE OBVIOUS IN VIEW OF PARSONS.....	59
	A. The Challenged Claims Are Obvious Over Parsons's FPGA- Implemented Order Book Server	62
	1. Discussion of Claim 43	62
	2. A POSA Would Have Been Motivated to Implement the OBC with LVC Memory Manager and Record/Message Updater	66

3. Claim 43	72
a. [43PRE] “An apparatus for applying specific computer technology to reduce latency and increase throughput with respect to streaming data enrichment”	72
b. [43A] “a coprocessor that includes an order engine and a price engine”	73
i. “a coprocessor...”	74
ii. “an order engine and a price engine”	75
c. [43B] “wherein the coprocessor is configured to receive streaming data representative of a plurality of limit order events pertaining to a plurality of financial instruments”	80
d. [43C] “wherein the order engine is configured to [43C.1] (1) access a plurality of limit order records based on the streaming limit order event data, [43C.2] (2) compute updated limit order data based on the accessed limit order records and the streaming limit order event data”	83
e. [43D] “wherein the price engine is configured to [43D.1] (1) access a plurality of price point records based on the streaming limit order event data, and [43D.2] (2) compute updated price point based on the accessed price point records and the streaming limit order event data”	92
f. [43E] “wherein the coprocessor is further configured to enrich the streaming limit order events with financial market depth data based on the computed updated limit order data and price point data”	99
g. [43F] “wherein the order engine and the price engine are configured to perform their respective operations in parallel with each other as the limit order event data streams through the coprocessor”	108
4. Claim 44	112
a. [44A] “comprises a member of the group consisting of a reconfigurable logic device, a chip-multi-processor (CMP), and a graphics processing unit (GPU)”	113
b. [44B] “wherein the order engine and the price engine are deployed on the member.”	113

B. The Challenged Claims Are Rendered Obvious By a Reconfigurable Logic Device Implementing Both Parsons’s OBC and price Summary LVC FAMs in the Same FAM Pipeline.....	113
1. Parsons’s Market Platform	114
2. A POSA Would Have Motivated to Implement Parsons’s OBS and Price Summary LVC on a Single “Ticker Plant” FPGA.....	117
3. Implementing the OBS on the Ticker Plant FAM Pipeline	122
4. Claim 43	125
a. [43PRE] “An apparatus for applying specific computer technology to reduce latency and increase throughput with respect to streaming data enrichment”	126
b. [43A] “a coprocessor that includes an order engine and a price engine”	126
c. [43B] “wherein the coprocessor is configured to receive streaming data representative of a plurality of limit order events pertaining to a plurality of financial instruments”	127
d. [43C] “wherein the order engine is configured to [43C.1] (1) access a plurality of limit order records based on the streaming limit order event data, [43C.2] (2) compute updated limit order data based on the accessed limit order records and the streaming limit order event data”	128
e. [43D] “wherein the price engine is configured to [43D.1] (1) access a plurality of price point records based on the streaming limit order event data, and [43D.2] (2) compute updated price point based on the accessed price point records and the streaming limit order event data”	130
f. [43E] “wherein the coprocessor is further configured to enrich the streaming limit order events with financial market depth data based on the computed updated limit order data and price point data”	134
g. [43F] “wherein the order engine and the price engine are configured to perform their respective operations in parallel with each other as the limit order event data streams through the coprocessor”	137
5. Claim 44.....	140

I, Bernard S. Donefer, declare:

1. My name is Bernard S. Donefer. I have been retained by Wolf, Greenfield & Sacks, P.C., counsel for ACTIV Financial Systems, Inc. (“ACTIV”), to submit this declaration in connection with a Petition for *Inter Partes* Review of claims 43 and 44 (the “challenged claims”) of U.S. Patent No. 10,062,115 (“the ’115 patent”).

2. I am being compensated for my time at a rate of \$700.00 per hour, plus actual expenses. My compensation is not dependent in any way upon the outcome of this matter and I do not have any financial interest in the outcome of this matter.

I. PERSONAL AND PROFESSIONAL BACKGROUND

3. I am an Adjunct Associate Professor at New York University’s Stern School of Business where I teach various courses related to financial information systems. I am a retired Distinguished Lecturer in the graduate Department of Information Systems and Statistics and Associate Director of the Wasserman Trading Floor - Subotnick Financial Services Center, at Baruch College of the City University of New York. I also previously taught at Fordham University, Graduate School of Business as Adjunct Assistant Professor in both the Finance and Information Systems Departments.

4. At all three institutions, I have taught graduate courses related to the use of technology in the financial services industry, including trading, market data, market structure and risk management. These include “IT in Financial Markets”, “Financial Information Systems” and “Risk Management Systems.”

5. I hold a BA in Economics from Long Island University and an MBA in Finance from NYU’s Stern Business School.

6. I am Principal of Conatum Consulting LLC. I teach both public and in-house corporate courses including, “Capital Markets Bootcamp”, “New U.S. Equity Markets” and “Risk Management for Non-Quants”. Clients included the Securities and Exchange Commission (SEC) and Federal Reserve Bank (FRB) in Washington, D.C., the Investment Industry Regulatory Organization of Canada (IIROC) in Montreal and Toronto, and corporate clients such as the Harvard Management Company, (university endowment), Depository Trust and Clearing Corp. (DTCC), ITG, BMO, JPMC, SEI, KPMG, et al.

7. I have published essays in the *Journal of Trading*, *Tabb Forum*, and *Bloomberg View*. I have been quoted in, among others, the *NY Times*, *Wall Street Journal*, *MIT Technology Review*, *The Atlantic Magazine*, *Reuters*, *American Banker*, *Businessweek*, *Wired*, and publications in Europe and Asia. I have been interviewed on CCTV (Chinese TV), NPR, BBC and Australian radio.

8. I have been an invited speaker at the Council of Foreign Relations in Washington D.C., The Conference Board in New York, committees of the NY Bar Association, and numerous industry events, including chairing duties at TradeTech's annual conference from 2010-2014.

9. I have more than 50 years of experience in the fields of software, technology and financial services. Prior to beginning my teaching career in 2003, I was Senior Vice President of Fidelity Investments in Boston, where I headed Capital Markets Systems. Previous positions include Executive Vice President and CIO of Dai Ichi Kangyo Bank in the U.S., then the world's largest bank by assets, President of BT Financial Services Information Systems, a subsidiary of Bankers Trust Co., where I was responsible for clients and staff in 14 cities in the U.S., Europe and Asia. As a consultant, my clients included the New York Stock Exchange and Milan Stock Exchange and over my career worked with asset classes including equities, fixed income, foreign currencies and derivatives. I have also been an expert witness for SIFMA, (broker/dealer, asset manager trade organization) in their SEC suit against the NYSE and NASDAQ for excessive pricing of depth of book data.

10. My technological experience extends back to the mid 1960's where I programmed early (2nd generation) IBM computers such as the 1620, 1401, 7040/90 in Fortran, machine and symbolic (SPS) languages.

11. I was hired by IBM in 1969 to work as an assembler language programmer for IBM's System Development Division, developing operating system components for the then new IBM 360 line of mainframes. Notably, I began as an experienced programmer given my prior programming experience, not going through their trainee program.

12. Over the next decade I worked as the lead systems programmer for Loeb, Rhoades & Co. a major brokerage house, then Senior Systems Officer and Head of Analytical Support to Management in the Treasurer's Division at Citibank. I managed programming tasks on IBM, DEC and Tektronix computers.

13. I next worked as a consultant at Coopers & Lybrand rising to manager in the financial industry technology practice. Client projects included the New York Stock Exchange, Milan (Italy) Stock Exchange, Lehman Bros. and numerous banks and brokers.

14. I next became President and CEO of CAP Information Systems, the US subsidiary of the UK based CAP Group Ltd. We specialized in systems development for banks and securities firms. At my direction, CAP became an early adopter of the UNIX operating system and C programming language in the financial services industry.

15. Later, as Director at IMNET, the IBM Merrill Lynch joint venture that sought to build a new market data system, I worked in product design and represented the organization in Europe.

16. As president of Bankers Trust Financial Information Systems, I was responsible for trading system development, on the DEC VAX platform, marketing and support for front office trading products at 40 client sites in the US, Europe and Asia.

17. I later became Executive Vice-President (EVP) and Chief Information Officer (CIO) of US technology operations at Dai Ichi Kangyo, then the world's largest bank by assets. After the 1993 World Trade Center bombing, I was responsible for Dai Ichi Kangyo's operational recovery, building a new trading room from scratch in three days.

18. I then became a Senior Vice President (SVP) and head of Capital Markets Systems at Fidelity Investments in Boston, where I lead the design and implementation of what we believe to be the first paperless, straight through processing sell side equity trading system. The system that I led the design of eliminated paper tickets, phone, fax orders and moved as early adopter of the FIX communications protocol. This project required maintaining management level expertise in C++ and object-oriented programming, bus technology alternatives, in a UNIX SUN server environment. The project even won the Fidelity President's

Award. Other key projects included managing the real time risk management system development and leading firm wide initiative to provide access to retail and institutional clients to the Redi electronic exchange (ECN).

19. Over my career I programmed, developed or managed systems on computers by IBM, DEC, Sun, Stratus and in languages that included Basic, Cobol, PL/1, APL, C, C++, Python, etc. and communications protocols as esoteric as the vertical blanking interval (VBI) in broadcast televisions signals.

20. My experience has earned me life membership in the Association of Computing Machinery, the world's largest educational and scientific computing society.

21. My curriculum vitae is provided as Ex. 1005.

II. MATERIALS REVIEWED AND CONSIDERED

22. My findings, as explained below, are based on my years of education, research, experience, and background in the field of financial information systems, software, technology, trading, exchanges, electronic trading and business practices in the financial markets, as well as my investigation and study of relevant materials for this declaration. In forming my opinions, I have studied and considered the materials identified in the list below.

Exhibit	Description
1001	U.S. Patent No. 10,062,115

Exhibit	Description
1002	File History for U.S. Patent No. 10,062,115 (“the ’115 FH”)
1003	U.S. Provisional Patent Application No. 61/122,673 (“the ’673 provisional”)
1006 ¹	U.S. Patent Publication No. 2008/0243675 (“Parsons”), now U.S. Patent No. 7,921,046
1007	U.S. Provisional Patent Application No. 60/814796 (“the ’796 provisional”)
1008	U.S. Patent No. 7,921,046 (“the ’046 patent”)
1009	U.S. Patent No. 6,278,982 (“Korhammer”)
1010	<i>Newton’s Telecom Dictionary</i> (21st Ed. 2005) (“Newton’s Telecom ’05”)
1011	<i>Newton’s Telecom Dictionary</i> (21st Ed. 2009) (“Newton’s Telecom ’09”)
1012	Revised Case Management Schedule; <i>Exegy Inc. v. ACTIV Financial Systems, Inc.</i> , Case No. 1:19-cv-02858, Dkt. No. 110 (N.D.IL).
1013	“XDP Integrated Feed Client Specification,” Version 2.3a, October 25, 2019 (“NYSE Spec”)
1014	Complaint; <i>Exegy Inc. v. ACTIV Financial Systems, Inc.</i> , Case No. 1:19-cv-02858, Dkt. No. 1 (N.D.IL).
1015	Foundry Networks Press Release, April 7, 2004
1016	Altera White Paper, October 2007

¹ Ex. 1004 is this declaration and Ex. 1005 is my CV.

III. MY UNDERSTANDING OF PATENT LAW

23. In developing my opinions, I discussed various relevant legal principles with ACTIV's attorneys. Though I do not purport to have prior knowledge of such principles, I understood them when they were explained to me and have relied upon such legal principles, as explained to me, in the course of forming the opinions set forth in this declaration. My understanding in this respect is as follows:

24. I understand that "*inter partes* review" (IPR) is a proceeding before the United States Patent & Trademark Office ("Patent Office") for evaluating the patentability of an issued patent claim based on prior art patents and printed publications.

25. I understand that in this proceeding Petitioner has the burden of proving that the challenged claims of the '115 patent are unpatentable by a preponderance of the evidence. I understand that "preponderance of the evidence" means that a fact or conclusion is more likely true than not true.

A. Claim Construction

26. I understand that in a patent, the "claims" define in technical terms the extent (i.e. the scope) of the protection conferred by a patent. I further understand that the first step in evaluating the validity of a claim is understanding each and

every term or phrase used in the claims, a process that is called “claim construction.”

27. I understand that in IPR proceedings claim language in a patent are given its plain and ordinary meaning consistent with the patent specification and prosecution history. If the specification provides a special definition for a claim term, that definition controls—even if it may be different from the meaning the term generally has in the field of the invention.

28. I have thus used the plain and ordinary meaning of the claim language found in the ’115 patent, consistent with how a person of ordinary skill in the art (“POSA”) at the time the invention was made would have understood the claim language unless I specifically explain that a term has a different meaning.

B. Prior Art & Priority

29. I understand the information that is used to evaluate whether a claimed invention is patentable is generally referred to as “prior art” and includes patents and printed publications (e.g., books and journal publications).

30. I understand that for an invention claimed in a patent to be patentable, it must be, among other things, new (novel) and not obvious from the prior art to a POSA at the time the invention was made.

31. I understand that determining whether or not a particular patent or printed publication constitutes prior art to a challenged patent can require

determining the priority date (also known as the effective filing date) of the challenged patent. In particular, I understand that for a challenged patent claim to be entitled to the benefit of the filing date of an earlier application (such as a provisional application) referenced in the challenged patent, the earlier application must adequately describe the invention that the applicant seeks to patent. In other words, I understand that the relevant question is whether the earlier application describes the claimed invention in sufficient detail for a POSA to reasonably conclude that the inventor had possession of the invention claimed in the issued patent as of the filing date of the earlier application. I also understand that a disclosure by an inventor that renders obvious a claimed invention is not sufficient to entitle the inventor to the benefit of the disclosure.

C. Obviousness

32. I understand that there are multiple ways in which prior art may render a patent claim unpatentable. I understand that one of those ways is that the prior art may have made the claim “obvious” to a POSA at the time the invention was made in view of a single prior art reference or a combination of prior art references.

33. I understand that a patent claim is obvious if the differences between the subject matter of the claim and the prior art are such that the subject matter as a whole would have been obvious to a person of ordinary skill in the relevant field at

the time the invention was made. Specifically, I understand that the obviousness question involves a consideration of:

- the scope and content of the prior art;
- the differences between the prior art and the claims at issue;
- the knowledge of a person of ordinary skill in the pertinent art;
- and
- whatever objective factors indicating obviousness or non-obviousness may be present in any particular case – referred to as “secondary considerations.”

34. I understand that in order for a claimed invention to be considered obvious, a person of ordinary skill in the art must have had a reason for combining teachings from multiple prior art references (or for altering a single prior art reference, in the case of single-reference obviousness) in the fashion proposed.

35. I further understand that in determining whether a prior art reference would have been modified in the case single-reference obviousness, combined with other prior art, or with other information within the knowledge of POSA, the following are examples of approaches and rationales that may be considered:

- combining prior art elements according to known methods to yield predictable results;

- simple substitution of one known element for another to obtain predictable results;
- use of a known technique to improve similar devices in the same way;
- applying a known technique to a known device ready for improvement to yield predictable results;
- applying a technique or approach that would have been “obvious to try,” i.e., choosing from a finite number of identified, predictable solutions, with a reasonable expectation of success.
- known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces if the variations would have been predictable to one of ordinary skill in the art;
- some teaching, suggestion, or motivation in the prior art that would have led one of ordinary skill to modify the prior art reference or to combine prior art reference teachings to arrive at the claimed invention. I understand that this teaching, suggestion or motivation may come from prior art reference or from the knowledge or common sense of one of ordinary skill in the art.

36. I understand that for a single reference or a combination of references to render the claimed invention obvious, a POSA must have been able to arrive at the claims by altering or combining the applied references.

IV. OVERVIEW OF RELEVANT TECHNOLOGY

A. Financial Markets and Financial Data

37. Financial markets and exchanges serve three fundamental purposes: (1) listing securities, offering access to liquidity; (2) receiving orders for, and trading in, securities, providing price discovery; and (3) providing information about those orders and trades to the public.

38. An exchange is a formal marketplace that brings together buyers and sellers of equities, futures and options contracts or other financial instruments, subject to rules intended to ensure that the trading activity occurs in an organized and fair manner. Orders are transmitted to the exchange, via its members, where they can be executed either directly between parties or intermediated via dealers. When exchanges were initially developed, and through much of their history, the placement of orders and execution of trades were manual, done by dealers, market makers and traders face-to-face or by phone. Today, most trading is done via computers and high-speed data networks, with orders directed using smart order routers and algorithms—automated computer programs designed to strategize and execute trades in the most profitable way possible.

39. The orders and trades placed by market participants on exchanges are a source of discovering the market price for a particular financial instrument. Exchanges aggregate and assimilate information received from customer orders and trades placed by broker-dealers and other members of the exchanges. Such information both reflects and drives investor expectations as to the value of those securities. Information regarding the value underlying a financial instrument, such as the earnings or other news about a company with respect to stock in that company, affects the supply and demand for the financial instrument, as reflected in customer orders. The supply and demand for the financial instrument, in turn, determines the price at which the financial instrument will trade.

40. Those wishing to buy or sell financial instruments provide instructions to exchanges known as orders. Orders to buy are known as bids and orders to sell are known as offers or asks. There are various order types, the most common being the market and limit orders.

41. With a market order, an investor, or broker acting on an investor's behalf, submits an order to buy or sell a financial instrument for immediate execution, naming the desired financial instrument and the quantity, to an exchange. A market order does not specify a price, but requests the best available price currently in the market. Market orders ensure timely execution, but at possibly unexpected prices, because prices can change in the fraction of a second

between the time when the trader sends the order and when that order is executed by the exchange.

42. Another type of order is a limit order. With a limit order, an investor, or broker acting on an investor's behalf, places either a "bid" in which she sets the highest price she is willing to pay to purchase a given financial instrument, or an "ask" or "offer" in which she sets the lowest price she is willing to accept in selling a given financial instrument. These bids and offers, when displayed publicly, are referred to as market quotes. In contrast to a market order, a limit order lets an investor wait for her desired price, but if there are no counterparties willing to sell or buy (as the case may be) at the limit price, the order will not be immediately fulfilled and, if the market price moves away from the limit price, the order may never be executed. The difference between the bid and ask prices is called the "spread." While waiting to be executed, these orders are held in an order book for that security at the exchange.

43. On an electronic exchange, as all 13 major U.S. equity exchanges are today, orders are submitted electronically via communications networks, using the FIX, FAST or similar standard protocols and are matched automatically to orders submitted by counterparties based on price, in order of arrival, by the exchange's matching engine software. When a market order is submitted, the exchange's computer system immediately executes the trade at the best available price. If the

best price is at a competing exchange, the receiving exchange must route the market order to the exchange with the best price. When a limit order is submitted, the exchange's computer system will execute the trade only if it can be filled at a price equal to or better than the price specified. In both cases, execution occurs either at the receiving exchange or after being routed to another exchange, if that is where the best price is available.

44. Providing market data regarding orders and executed trades on a timely basis is critical, especially as, at least in the U.S., there are rules requiring orders are executed at the best possible price. That is, the investor must receive the best available price in the market, even when there are multiple competing exchanges trading the same financial instrument.

1. Order Books

45. An order book is a data file, sorted by price and time of entry, of limit orders for each financial instrument that have been received and entered for execution, but not yet fulfilled. Electronic exchanges maintain an order book for each financial instrument traded on that exchange electronically. To facilitate trading strategies (e.g., for buying or selling shares of stock), it was conventional for traders (e.g., trading firms), when such data became available, to construct and maintain their own order book that mirror the order books on one or more exchanges, providing depth of market information.

46. Order books were conventionally updated from feeds of data provided by the financial exchanges which disseminate limit order “events” (e.g., addition, modification or deletion of limit orders). Speed is important in updating a trader’s order book as new order events are continuously received because traders (e.g., those engaged in electronic trading) often seek to exploit market opportunities before they quickly dissipate. Parsons explains, for example, “speed of information delivery can often translate into actual dollars and cents for traders, which in large volume situations, can translate to significant sums of money.” Parsons, [0005].

47. The term “top of book” or “inside market” refers to the highest limit order bid to buy and lowest limit order offer to sell, with their respective quantities. Said differently, the inside market is the best bid and offer order, sometimes called the “BBO,” in the exchange’s order book. The best bid and offer orders across all markets is known as the national best bid and offer or “NBBO.”

48. Figure 1, below, depicts a market quote for Microsoft stock, and the portion outlined in red shows the top of book or inside market and is the NBBO for Microsoft stock (“MSFT”). As seen in the red box, the current highest bid for Microsoft stock is \$57.41 for a total of “40,” which is multiplied by 100 to get the number of shares, meaning the bid is for 4,000 shares. As also seen, the current lowest offer (or ask) is \$57.72 for a total of “26,” or 2,600 shares (Note that the

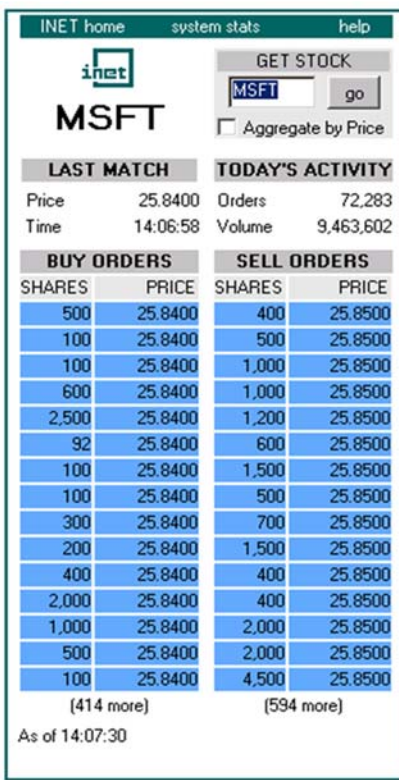
total number of shares may be made up of one or more individual orders at that price.). The spread, the difference between the best bid (\$57.41) and the best ask (\$57.42) is one cent. The market quote for Microsoft stock in Figure 1 has been enriched with other fields including total volume traded; high, low price for the day; and VWAP, etc. These values are all calculated from the order book and from historical trades as well as fundamental data such as dividend.

Figure 1: Exemplary Market Quote For Microsoft Stock

Microsoft Corporation										Fin'l Status: N				NASDAQ	
57.42 + 0.02 0.03%				Bid - Ask		52Wk H	58.70	EPS	2.1	P/Sales	NYSE MK	Off-Hr Price	57.59		
				57.41 57.42		52Wk L	43.66	P/E	27.34	Shares(m)	7792.52	Off-Hr Vol	50		
Vol	100	CVol	5,591,837	40 x 26		3M PChg	12.20%	FY1 EPS	2.90	MktCap	Chicago	Off-Hr Cvol	18403		
Time	10:40:1	30D Avg	22,475,560	10:40:16a 10:40:16a		12M PCh	29.72%	FY1 P/E	19.81	Dividend	0.39	Off-Hr Time	09:30:00a		
Exch	ADF	Vol Pct	24.88%	PHL PHL		High	57.68	Bk Val PS	9.22	Div Yield	NASDAQ	Pre %Chg	0.33%		
Open	57.57	Block Qty	10	VWAP		57.45	Low	57.34	P/Bk Val	6.23	ExDate	US Option	Post %Chg	0.30%	
PrClose	57.40	Block Vol	541218	Beta		1.23	Packaged Software			Headlines		16	REALTIME	USD	

49. Figure 2, below, depicts an example of an order book for Microsoft stock from the INET electronic exchange in approximately 2003, and shows each order for Microsoft stock, including both the number of shares and the price. The orders are listed in the order in which they were received. The inside price at INET as shown in Figure 2 would be a bid of \$25.84 and an offer of \$25.85, with a one cent spread. The order book in Figure 2 has also been enriched with historical activity that had occurred already in that day, including total volume and last price information. The INET electronic exchange would be an example of a regional exchange, in the parlance used by the '115 patent.

Figure 2: Exemplary Order Depth View For Microsoft Stock



50. Figure 2 would be a view supported by the “full order depth” order book construction illustrated in, and comparable to, Figure 2(a) of the ’115 patent and discussed in the background of the ’115 patent. Figure 2, above, also has a “Aggregate by Price” check box that allows a trader to switch to a price-aggregated view that would have been supported by the “price aggregated depth” order book construction illustrated in Figure 2(b) of the ’115 patent, and as discussed in the background of the ’115 patent. Figure 2, above, and Figure 2(b) of the ’115 patent both would have been a commonly available display since the late 1990’s.

51. Figure 3, below, depicts a different example of an order book for Microsoft stock, but this time depicts data from both the Arca and the INET exchanges taken prior to the March 8, 2006, acquisition of Arca by the NYSE (INET was acquired by the NASDAQ market). Figure 3 shows the aggregate order interest at each price level. Figure 3 would be an example of a spliced priced summary view, combining regional exchange order books, in the parlance used by the '115 patent.

Figure 3: Exemplary Price Aggregated View For Microsoft Stock

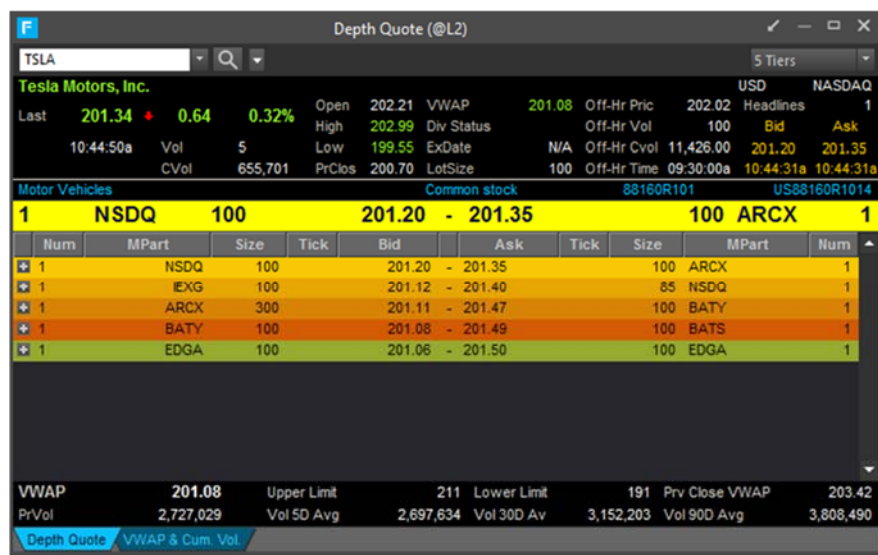
Bid				Ask			
ID	Price	Size	Time	ID	Price	Size	Time
ARCAEX	25.83	38925	14:11:00	ARCAEX	25.84	39892	14:11:00
INET	25.83	33400	14:10:57	INET	25.84	40047	14:11:00
ARCAEX	25.82	52590	14:11:00	ARCAEX	25.85	48200	14:10:56
INET	25.82	60643	14:10:28	INET	25.85	44235	14:10:31
ARCAEX	25.81	67200	14:10:47	ARCAEX	25.86	50932	14:10:43
INET	25.81	90260	14:10:56	INET	25.86	36241	14:10:57
ARCAEX	25.80	42600	14:10:57	ARCAEX	25.87	38432	14:10:56
INET	25.80	42270	14:10:47	INET	25.87	36185	14:10:41
ARCAEX	25.79	23600	14:10:57	ARCAEX	25.88	19900	14:10:42
INET	25.79	25200	14:10:56	INET	25.88	37449	14:10:57
ARCAEX	25.78	18800	14:10:50	ARCAEX	25.89	17800	14:09:36
INET	25.78	18700	14:10:42	INET	25.89	20723	14:10:41
ARCAEX	25.77	23900	14:10:50	ARCAEX	25.90	26000	14:10:00
INET	25.77	12848	14:10:42	INET	25.90	17040	14:10:37
ARCAEX	25.76	20900	14:08:51	ARCAEX	25.91	14300	14:10:00
INET	25.76	27200	14:09:49	INET	25.91	12800	14:09:48

Connected to Arca Market Data server.

52. Figure 3 is comparable to Figure 4(b) of the '115 patent. The Arca exchange would be another example of a regional market, in the parlance used by the '115 patent, along with the INET exchange.

53. Figure 4, below, depicts the national best bid and offer, 210.20 – 201.35 respectively, for Tesla stock. The best bid and offer is calculated from the order book for the stock, aggregated across all of the regional markets – five in the example depicted by Figure 4. As can be seen, the best bid and the best offer may be available at different regional markets.

Figure 4: Exemplary National Best Bid And Offer For Tesla Stock



54. Figure 4 is also comparable to Figure 4(b) in the '115 patent. If Figure 4, above, were to have depicted bid or offer depth below the best prices at each market, it would be an example of national depth of market.

2. Electronic Exchanges Electronic Data Feeds

55. Exchanges using electronically maintained order books were common well before 2000. Market data system vendors (e.g., Reuters, Bloomberg, etc.), brokers, and asset managers with their own ticker plants would rely on exchanges

sending continuous streams of data broadcasting orders, updates and executed trades. Ex. 1013 provides the specification for the data feed that the New York Stock Exchange streams.

56. The 115 patent defines the term “financial market data” as “data contained in or derived from a series of messages that individually represent a new offer to buy or sell a financial instrument, an indication of a completed sale of a financial instrument, notifications of corrections to previously-reported sales of a financial instrument, administrative messages related to such transactions, and the like.” ‘115 patent, 1:63-2:3.

57. The term “financial market depth data” often refers to financial market data that has been derived from financial market data, such as, volume, volume weighted prices, daily high, low, price changes, moving averages, tick direction, option pricing model outputs such as the “greeks”, e.g., delta, gamma, etc., or other data that, derived or not, reflects the current state of the financial market for a particular security, or more generally. Broadly, it can be any data reflecting the state of a financial instrument or the state of the financial market.

58. The ’115 patent uses the term “financial market depth data” broadly to describe the information in the *input stream* from the market exchanges (’115 patent, 5:15-31, 9:54-63), like it uses the term “financial market data.” The ’115 patent, other than referring to it as an input, does not otherwise define or describe

this term. Thus, the ‘115 patent uses the term broadly to describe any of a wide variety of market information, including information on outstanding limit orders of a financial instrument on the market (e.g., the number and price of open buy and sell orders for a financial instrument), trades, etc., and is therefore not limited to derived values since it also includes the information from the exchanges.

59. U.S. patent no. 6,278,982 (“Korhammer”) issued August 21, 2001. Ex. 1009, 1. Korhammer explains that computerized trading networks each maintained a central order book that “keeps track of bid/offer information including price, volume, and execution for each open and closed transaction as supplied to it in real time by its members.” Ex. 1009, 2:34-37. Korhammer discloses a client-based computer system that “aggregates order book information from each participating ECN order book computer including security, order identification, and bid/offer price information.” *Id.*, 4:28-31.²

60. Korhammer discloses that the “combined information is displayed to customers for the selected security separately for bids and offers, and sorted by price, volume and other available attributes as desired by the customer.” *Id.*, 4:31-36. Korhammer continues, (CCS being the consolidating computer system):

² I have used the abbreviation “*id.*”, which is short for the Latin word “idem,” meaning “the same” to refer to the same last exhibit previously cited.

“Once the information from a number of ECNs and electronic exchanges are combined in the CCS [Korhammer’s system], either the CCS, the trading terminal, or both can be used to calculate real time metrics. The real time metrics, such as volume trends, price trends, and various on demand calculations, can aid the trader in making decisions. The CCS can also determine the occurrence of market events in which its customers may be interested, such as a new high bid for the day or a locked market where the best bid is equal to the best offer.”

Id., 4:46-54.

61. Korhammer explains precisely how the order book and depth of market data is reconstituted in Fig. 6, which is a flow diagram showing how the consolidated order book is formed. *Id.*, 9:27-49.

62. Korhammer provides exemplary ways in which the reconstituted order book and market depth data can be displayed, such as in the ’982 patent’s Figure 4, reproduced below, showing a “spliced” order book:

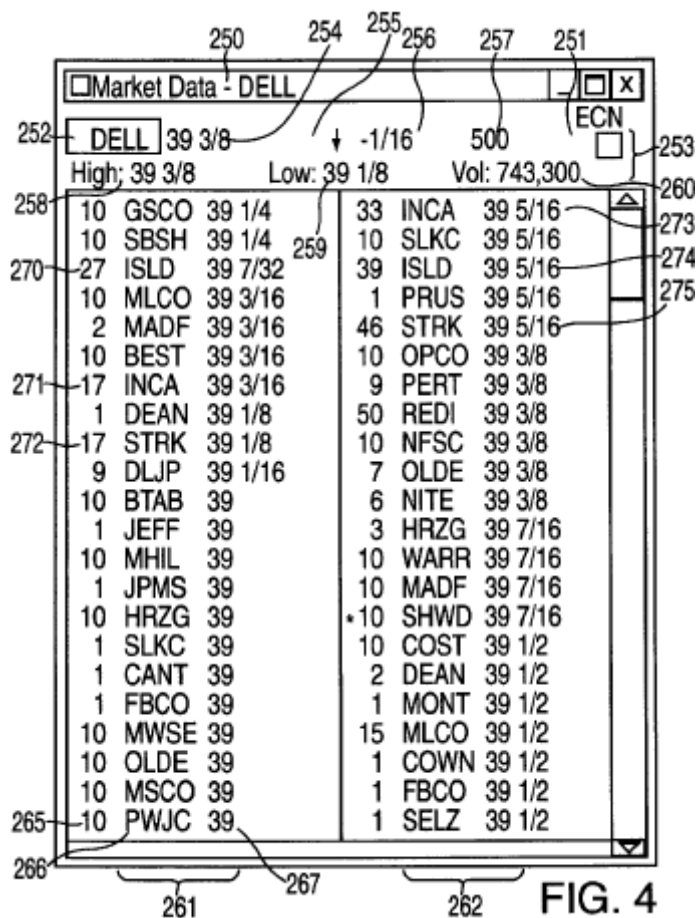


FIG. 4

Korhammer explains that its Figure 4, above, represents a typical market data screen that displays both primary and secondary, market depth, data. *Id.*, 8:47-9:8.

63. Korhammer was the basis for the very successful Lava product, which took order book data from multiple exchanges, created a single view of the market, enabling asset managers and brokers to route their orders to the best prices and sources of the greatest liquidity, in the least amount of time. It could also be used to create added value products by incorporating proprietary data into its display. In April 2004 Lava used communications coprocessing technologies to improve its performance. “Lava Trading Inc., a leading developer of high-performance trading

solutions for the financial services industry, has deployed Foundry's ServerIron® 400 Layer 4-7 load balancing switches to increase capacity and availability of its high-performance financial trading system." *See* Ex. 1015.

B. Coprocessor Technology

64. Since the invention of the microprocessor, processor performance has doubled every 18 months or so, a trend known as "Moore's Law." Each new round of processor improvements made more complex and demanding applications achievable, but also continued to expand user expectations for further improvements. For some applications, the need for ever faster data processing outstripped the available performance in existing microprocessor technology, creating a gap between the demand for high speed processing and the existing performance.

65. Once such example is in financial analytics, where financial trading houses seek to gain a competitive advantage in the market by accelerating their quantitative applications such as options valuation, statistical arbitrage models, algorithmic trading, market making models, etc. Financial application acceleration enables the financial trading houses to make trades faster and with more accuracy than the competition, resulting in financial gains for the trading house. Achieving the necessary performance, however, requires substantially increasing processing

power to meet or exceed competitor's capabilities. Opportunities in the market are taken by the first entity to identify them.

66. Fortunately, increasing processor performance is not the only way to address rising processing demands by applications. Enhancing communications bandwidth and disk speed are two conventional approaches. Further, augmenting a processor with an application-specific coprocessor has long proven to be a viable approach to solving performance shortfalls.

67. A coprocessor in the art is a computational engine designed to operate in conjunction with other components in a computational system having a main processor. The coprocessor is optimized to perform a specific set of tasks and is used to offload tasks from a main processor in order to optimize system performance. In 1984, when I configured my first IBM PC, I added a Hercules card, which was a graphics coprocessor, enabling my machine to display higher resolution graphics. Interestingly, the graphics chip could be re-programmed to further enhance processing. These speed enhancing additions had a long and well known history.

68. Use of such coprocessors began with specialized I/O processing such as modems and Ethernet controllers, extending to the use of graphics rendering engines as display demands rose and encryption engines for increased security. More general-purpose coprocessors also arose, first in the form of math

accelerators to handle multiplication and division. For example, on a I386 chip PC, I could purchase an Intel 8087 math co-processor for a five-fold improvement in math problem processing speeds. Digital signal processors then arose as coprocessors designed to handle complex mathematical algorithms by incorporating built-in math hardware and employing new architectures that featured pipeline and parallel structures.

69. Coprocessors can be of a number of various types. One type of coprocessor is known as an “Application-Specific Integrated Circuit” or “ASIC.” An ASIC is a coprocessor whose programming is determined at the time of manufacture and can never be changed once the chip is made. These are often used in the financial services industry by quantitative high frequency traders to implement their proprietary trading algorithms. The ’115 patent define coprocessor as “a computational engine designed to operate in conjunction with other components in a computational system having a main processor (wherein the main processor itself may comprise multiple processors such as in a multi-core processor architecture).” ‘115 patent, 2:59-63. I have applied this definition throughout. The ‘115 patent also states that: “Typically, a coprocessor is optimized to perform a specific set of tasks and is used to offload tasks from a main processor (which is typically a GPP) in order to optimize system performance. The scope of tasks performed by a coprocessor may be fixed or variable, depending on the

architecture of the coprocessor. Examples of fixed coprocessor architectures include Graphics Processor Units which perform a broad spectrum of tasks and floating point numeric coprocessors which perform a relatively narrow set of tasks. Examples of reconfigurable coprocessor architectures include reconfigurable logic devices such as Field Programmable Gate Arrays (FPGAs) which may be reconfigured to implement a wide variety of fixed or programmable computational engines. The functionality of a coprocessor may be defined via software and/or firmware.” ‘115 patent, 2:63-3:11.

1. Field Programmable Gate Arrays (FPGAs)

70. A “Field Programmable Gate Array” or “FPGA” is a different type of coprocessor than an ASIC because the programming of an FPGA can be changed after it is manufactured. The ability to reprogram FPGAs allows them to be made without any program-specific tooling, and allows for the programming to be updated after the chip is made.

71. FPGAs also have on-board memory. The on-board memory means that the coprocessor logic's memory access bandwidth is not constrained by the number of I/O pins the device has. Further, the memory is closely coupled to the algorithm logic and reduces the need for an external high-speed memory cache.

72. Performance of computer architectures using FPGAs have shown a ten-fold to hundred-fold improvement in the execution speed of algorithms as

compared to processors alone based on an October 2007 white paper by Altera (Ex. 1016), reproduced below:

Application	Processor Only	FPGA Coprocessing	Acceleration Factor
Hough and Inverse Hough Processing (1)	12 minutes processing time Pentium 4-3 GHz	2 seconds of processing time at 20 MHz	370X
Spatial Statistics (Two-Point Angular Correlation Cosmology) (2)	3,397 CPU hours with 2.8-GHz Pentium (approximate solution)	36 Hours (exact solution)	96X
Black-Scholes (Financial Application (single precision FP 2M points)) (2)	2.3M experiments/sec with a 2.8-GHz Processor	299M experiments/sec	130X
Smith Waterman SS search 34 from FASTA (1)	6461 sec processing time (Opteron)	100-sec FPGA processing	64X
Prewitt Edge Detection (compute intensive video and image processing) (3)	327M clocks (1-GHz processing power)	131K clocks at 0.33 MHz	83X
Monte Carlo Radiative Heat Transfer (1)	60-ns processing time (3-GHz processor)	6.12 ns of processing time	10X
BJM Financial Analysis (5M paths) (1)	6300 sec processing time (Pentium 4-1.5-GHz)	242 sec of processing at 61 MHz	26X

Notes:

- (1) Source: Celoxica
- (2) Source: SRC Computers
- (3) Source: XtremeData

73. As of at least 2006, FPGAs had the support of many mature development tools, such as C-to-hardware tools that converted programming in the C language to code suitable for FPGA programming. Code analysis tools were available to take a user's C code and identify functions that would benefit from hardware acceleration. And compilers existed to automatically structure the object code for these functions for parallel and pipelined execution to maximize the effectiveness of acceleration. Ex. 1016, 4-5.

74. The availability of this full design tool chain meant that users did not have to struggle to accelerate their existing applications, and allowed users to program FPGAs who did not have any hardware expertise. These tools also allowed users to use FPGAs without needing to re-write their source code for

coprocessing. This scenario was ideal for applications, such as financial services, where the software application functions are tightly regulated and changes involve costly or time-consuming re-testing.

75. FPGAs also have a significant computing power advantage over processors because of their parallel processing capabilities and internal bus architectures. This allows the FPGA to execute functions in only a few clock cycles instead of the hundreds to thousands of clock cycles that the sequential operation of a processor would require. Ex. 1016, 5.

76. FPGAs have a power advantage over processors as well. Because so few clock cycles are required for the same processing capability, FPGAs can operate with much slower clocks and still provide a performance boost. The lower clock speeds result in lower power consumption, making an FPGA coprocessor much more power-efficient than a processor. Ex. 1016, 5.

77. The cost of an FPGA coprocessor is comparable to, and often less than, that of a processor with similar performance. As a result, the component cost of a processor and FPGA coprocessor is no more than that of two processors in a standard cluster design.

V. SUMMARY OF THE '115 PATENT

78. The '115 patent is directed to high-speed processing of financial market data to maintain an up-to-date list of all outstanding offers to buy and sell

financial instruments (e.g., financial securities such as stocks, futures, etc.)—from one or more electronic exchanges referred to as the “order book” for the corresponding financial instrument. ’115 patent, 3:50-61.

79. As explained above in the background section, financial exchanges (e.g., NYSE, NASDAQ, etc.) maintain their own order books for all orders that flow into the exchange, and disseminate order events related to those orders (e.g., as an order add, modify or delete event) via electronic “market data feeds” to market data systems subscribing at the exchange. These market data systems process the order events received from the exchanges to maintain their own order books to keep track of the current state of financial instruments on the market. *Id.*, 3:50-61.

80. The ’115 patent’s background describes two examples of data structures (shown in Figs. 2(a)-(b) reproduced below) conventionally included in an order book. Fig. 2(a) shows a “full order depth” data structure that aggregates and lists all of the outstanding orders for a financial instrument (e.g., stock). *Id.*, 4:3-6. Fig. 2(b) shows a “price aggregated depth” data structure that shows the number of outstanding orders for a financial instrument at each price, thereby demonstrating the “distribution of liquidity (available shares) over prices for a given financial instrument.” *Id.*, 4:7-10. Both the full order (Fig. 2(a)) and price aggregated (Fig. 2(b)) data structures are organized into “ask side” orders (offers to

sell) and “bid side” orders (offers to buy), and sorted by price on each “side” of the order book.

					order #	price	time	size	
Sell order 339973 \$25.42 09:51:03 250					329091	\$25.43	09:47:23	50	Ask side
					328432	\$25.43	09:46:31	25	
					329881	\$25.42	09:51:03	100	
					329880	\$25.41	09:51:02	5	
									Top of Book
					328128	\$25.40	09:45:31	20	Bid side
					328127	\$25.40	09:45:31	10	
					329192	\$25.40	09:48:20	500	
					330821	\$25.39	09:58:23	200	
					329012	\$25.38	09:47:01	50	

Figure 2(a)

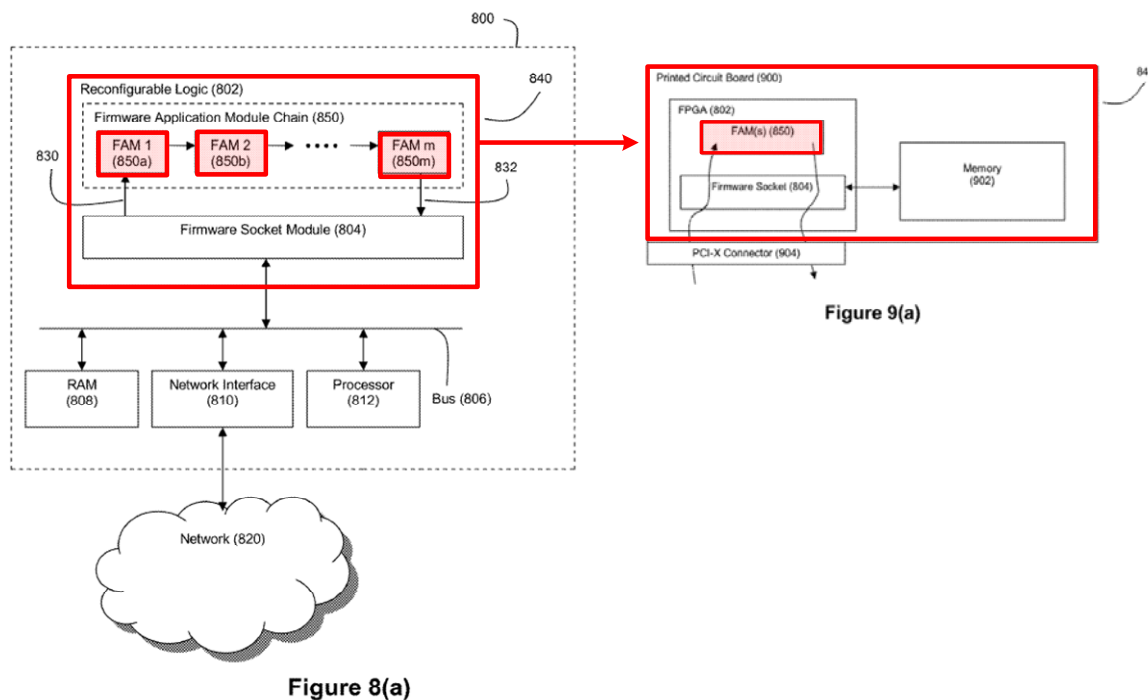
					order count	price	time	size	
Sell order 339973 \$25.42 09:51:03 250					5	\$25.45	09:47:23	540	Ask side
					2	\$25.43	09:46:31	75	
					1	\$25.42	09:51:03	100	
					1	\$25.41	09:51:02	5	
									Top of Book
					3	\$25.40	09:45:31	530	Bid side
					1	\$25.39	09:59:23	200	
					1	\$25.38	09:47:01	50	
					10	\$25.36	09:45:31	1500	
					9	\$25.35	09:48:20	1240	

Figure 2(b)

81. The '115 patent concedes that constructing and maintaining order books was known. *Id.*, 3:38-59, 4:63:5:1.

82. The purported advance claimed in the challenged claims (claims 43 and 44) is to accelerate order book updating by using a “coprocessor” to “offload” processing involved in updating order books to “decrease processing latency for those tasks relative to the main processor.” *Id.*, 2:58-3:17, 5:11-14. This coprocessor can be implemented using reconfigurable logic and, more specifically, Field Programmable Gate Arrays (FPGAs) that are configured to process financial market data at “hardware speeds” to perform order book maintenance with high throughput; orders to buy or sell a specified number of shares of a security (e.g., stocks, futures, options, etc.) at a specified price—called a “limit order”—that are traded on an electronic financial exchange (e.g., NYSE, NASDAQ, etc.). *Id.*, 3:55-58.

83. The '115 patent describes implementing this FPGA-based coprocessor by programming different firmware application modules (FAMs) to perform specific tasks. *Id.*, 10:13-11:42. Once programmed, FAM modules can be loaded onto reconfigurable logic (e.g., an FPGA) in a pipeline to achieve desired functionality, as illustrated by FAM chain 850 and its FPGA implementation illustrated in Figures 8(a) and 9(a) below. *Id.*



84. Figure 12(a), reproduced below, illustrates one example FAM pipeline configured to receive a stream of financial market data events pertaining to limit orders (i.e., limit order events), process the limit order events to update an order book for each instrument, and update the limit order events based on this processing. *Id.*, 5:18-31.

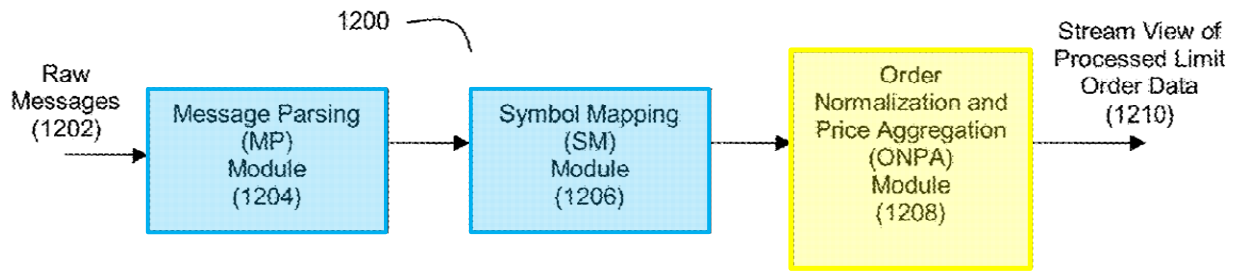


Figure 12(a)
(Annotated)

85. Each FAM module performs a specified operation on the stream of input messages from the exchanges that are passed to the next module in the pipeline. Specifically, input FAM modules highlighted in blue include a Message Parser (MP) Module 1204 that parses the received data stream and formats the information to provide consistent limit order event messages understandable by downstream modules (*id.*, 12:24-43), and Symbol Mapping (SM) Module 1206 that maps information in the limit order event messages to internal identifiers for use in addressing records in memory (*id.*, 12:44-60, 14:65-15:7).

86. The parsed limit order event messages and internal identifiers are passed to Order Normalization and Price Aggregation Module (ONPA) 1208 (highlighted in yellow) to access and update the financial instrument records representing the order books stored in memory based on the received limit order events (*id.*, 17:52-18:24). Specifically, the OPNA receives the stream of limit order events and updates corresponding limit order records (an example limit order record is illustrated in Fig. 20) and normalizes the input message based on the

received limit order events. *Id.*, 17:52-18:24. The stored limit order records correspond to the “full order depth” data structure illustrated in Fig. 2(a) in the background.

87. The ONPA may additionally maintain price point records to support price aggregated views the order book. *Id.*, 18:6-20. In this data structure, a record is maintained for each unique price and includes fields for the total number of shares and the total number of orders at each price point. *Id.* These “price aggregated data structures (*id.*, 20:26) correspond to the “price aggregated depth” data structure shown in Fig. 2(b) in the background.

88. The ’115 patent states that in a preferred embodiment, the order engine that updates the order book’s full order data structure and the price engine that updates the order book’s price aggregation data structure update their respective data structures in parallel within the same coprocessor. *Id.*, 20:25-32 (“parallel engines update and maintain the order and price aggregation data structures in parallel”). The ’115 patent refers to the one or more update engines that update and maintain the respective data structures as “order engines” and the “price engines,” respectively. *Id.*

89. The ONPA may be additionally configured to update the limit order events based on price aggregation computations by appending price attributes to the limit order events before providing the updated message streams to

downstream subscribers. *Id.*, 18:6-20, 23:41-44. As shown in Figure 12(c) below, additional output modules (also highlighted in blue) can be added to the FAM pipeline to filter and format the updated limit order event messages for downstream consumers (e.g., via Interest and Entitlement Filtering (IEF) Module 1210 and Message Formatting (MF) Module 1212) in accordance with each consumers subscription and format expectations. *Id.*, 23:41-24:5.

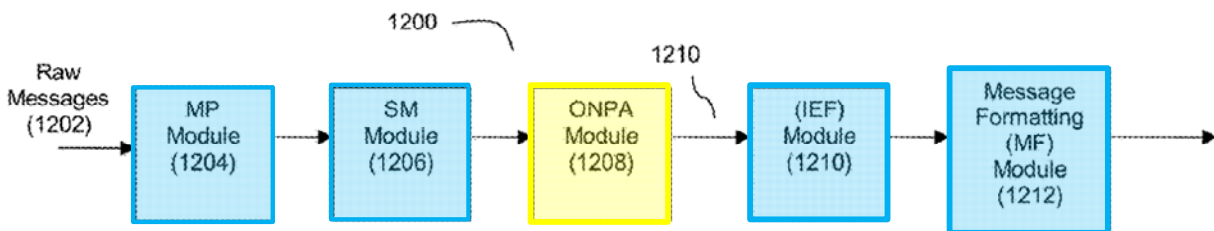


Figure 12(c)
(Annotated)

90. The '115 patent states that the Message Parser, Symbol Mapping, Interest and Entitlement Filter, and Message Formatting Modules are the same as those described in U.S. application pub. no. US 2008/0243675 A1 (“Parsons”). *Id.*, 12:35-44, 23:47-24:5; *see also* the incorporated ‘673 provisional (Ex. 1003), 31 (“[t]he message parser, symbol mapping and message formatting stages are described in U.S. Patent Application Publication 2008/0243675”), 34 (“[t]he Interest Entitlement Filter [is] described in the 2008/0243675”).

91. The ONPA Module 1208 is merely an obvious implementation of the caching modules disclosed in Parsons, one of which—the “order book cache

update, retrieve and normalize (OBC)” module (Parsons, [0071]) performs the same functionality as the ’115 patent’s ONPA module on which claims 43 and 44 are drawn.

92. The ’115 patent explains that the coprocessor having the ONPA module is employed at a “ticker plant”—a system designed to handle throughput of high-volume financial data for traders. ’115 patent, 3:38-5:14.

A. Level of Ordinary Skill in the Art

93. I am informed that prior art references should be understood from the perspective of a POSA to which the patent is related, based on the understanding of that person at the time of the patent’s priority date. I understand that a POSA is presumed to be aware of all pertinent prior art and the conventional wisdom in the art, and is a person of ordinary creativity. I have applied this standard throughout my declaration.

94. It is my opinion that a POSA would have had (1) a bachelor’s degree or higher in electrical engineering, computer engineering, or a similar field; (2) at least two years of experience working with financial computing systems; and (3) experience with systems having high throughput data streaming and processing requirements, for example systems for processing data feeds, such as the prior art system shown Fig. 1 of the ’115 patent. Additional education could have

substituted for professional experience, and significant work experience could have substituted for formal education.

95. I qualify as at least a person of ordinary skill in the art given my decades of experience in the financial and information technology industries. *See* section I, above. My opinions set forth herein are from the perspective of a POSA, as defined in the previous paragraph.

B. Overview of the Challenged Claims

96. I explain the challenged claims below at ¶¶ 150-157 (claim 43) and ¶ 268 (claim 44).

VI. DISCLOSURE OF PARSONS

97. U.S. application pub. no. US 2008/0243675 A1 (“Parsons”) published on October 2, 2008, and is the publication of application no. 11/765.306, which was filed on June 19, 2007. I am informed and understand that based on its filing and publication date and because it listed different inventors than the ’115 patent, Parsons is prior art under 35 U.S.C. §§ 102(a) and 102(e). Based upon my review of the file history for the ’115 patent, I understand that Parsons was considered during prosecution but only for dependent claims that are not being challenged.

A. Parsons Described the Same Reconfigurable FPGA’s for Performing the Same Processing of Financial Market Data Prior to the ’115 Patent

98. Parsons, titled “High Speed Processing of Financial Information Using FPGA Devices,” is directed to consolidating financial market data

operations conventionally performed by distributed networked computer systems to “reconfigurable logic, such as Field Programmable Gate Arrays (FPGAs)” to “process streaming financial information at hardware speeds.” Parsons, [0010]-[0011]. By consolidating this functionality on FPGAs, processing latencies can be reduced. *Id.*

99. Parsons discloses that conventional market data platforms distributed a wide variety of financial market data processing over a network of individual computers as shown by the numerous functional units 102 illustrated in Fig. 1, reproduced below. *Id.*, [0006].

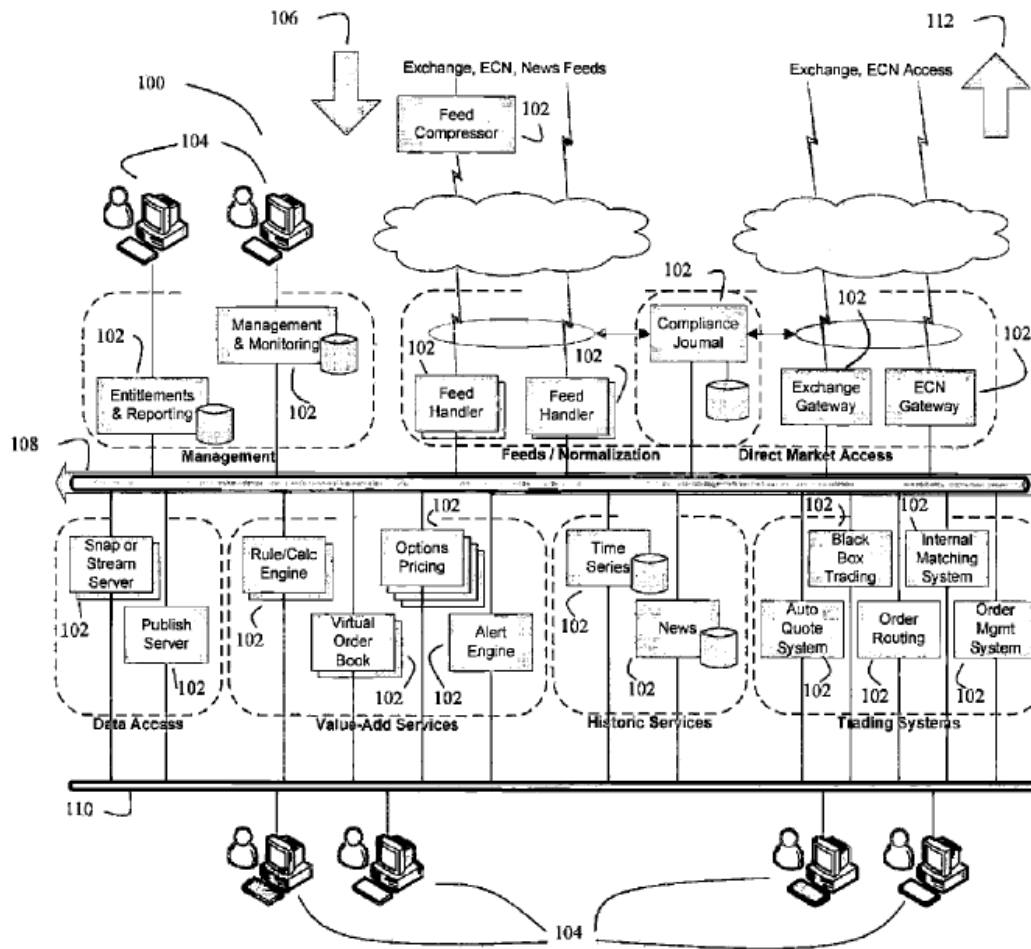
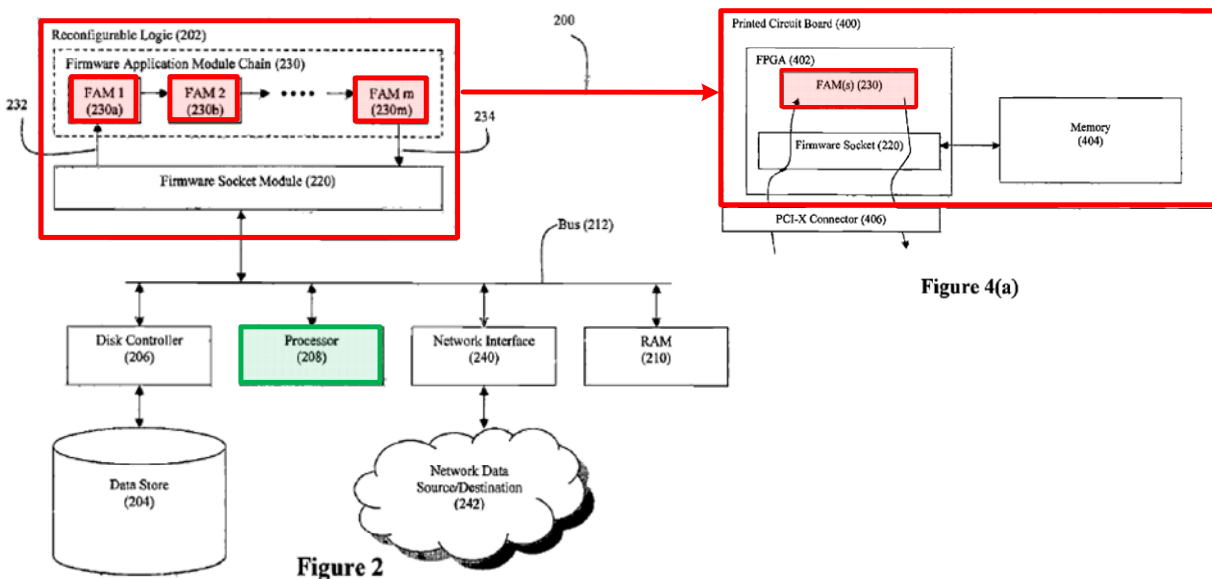


Figure 1
PRIOR ART

100. Parsons's explains that conventional market data platforms that attempted to deliver financial market data to traders in "real time" were implemented in software over a network of distributed individual computer systems using general purpose processors. *Id.*, [0006]-[0008]. In these systems, latencies were introduced by software execution and as a result of the overhead in transmitting messages to and from different machines. *Id.*, [0006]-[0008].

101. Parsons discloses a hardware-accelerated platform that “offloads” the same functionality onto a fewer number of reconfigurable logic devices. *Id.*, [0056]. To implement this functionality on reconfigurable logic (e.g., an FPGA), individual firmware application modules (FAMs) are programmed to perform specific data processing tasks. Once programmed, each FAM provides a “hardware template” (*id.*, [0057], [0068], [0077], [0083]) that performs a set of programmed tasks that can be loaded onto a reconfigurable logic device and chained together in a pipeline to perform desired data processing functions, as shown in Figure 2 reproduced below. *Id.*, [0045].

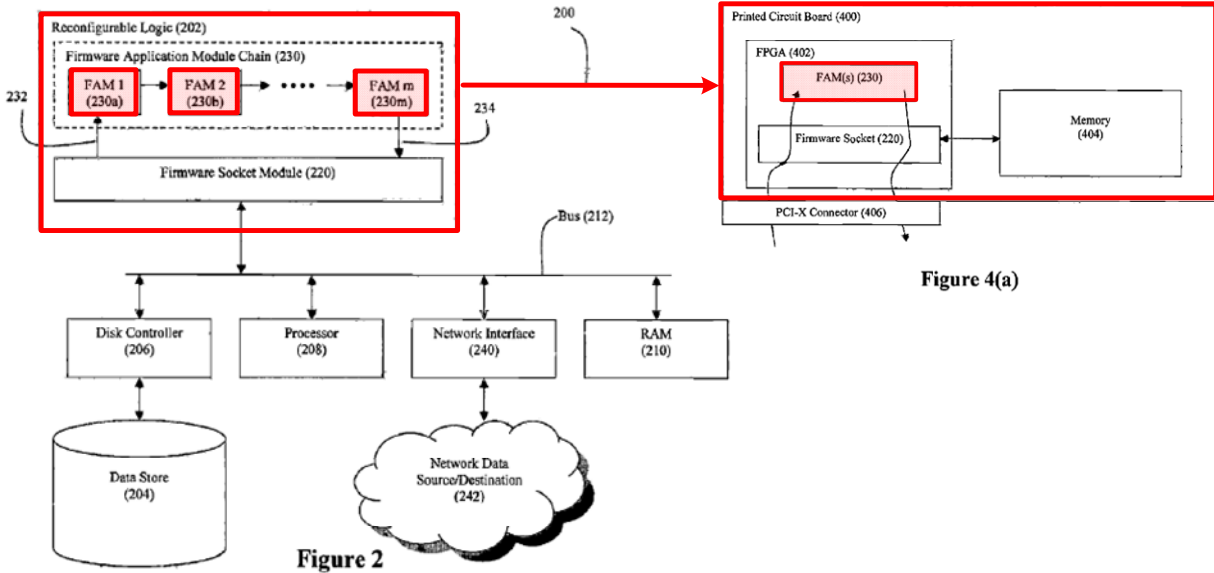


102. Parsons’s reconfigurable logic devices are described as being implemented on FPGA printed circuit boards that can be connected to “a system’s main processor 208” (highlighted in green above) to offload computations from the

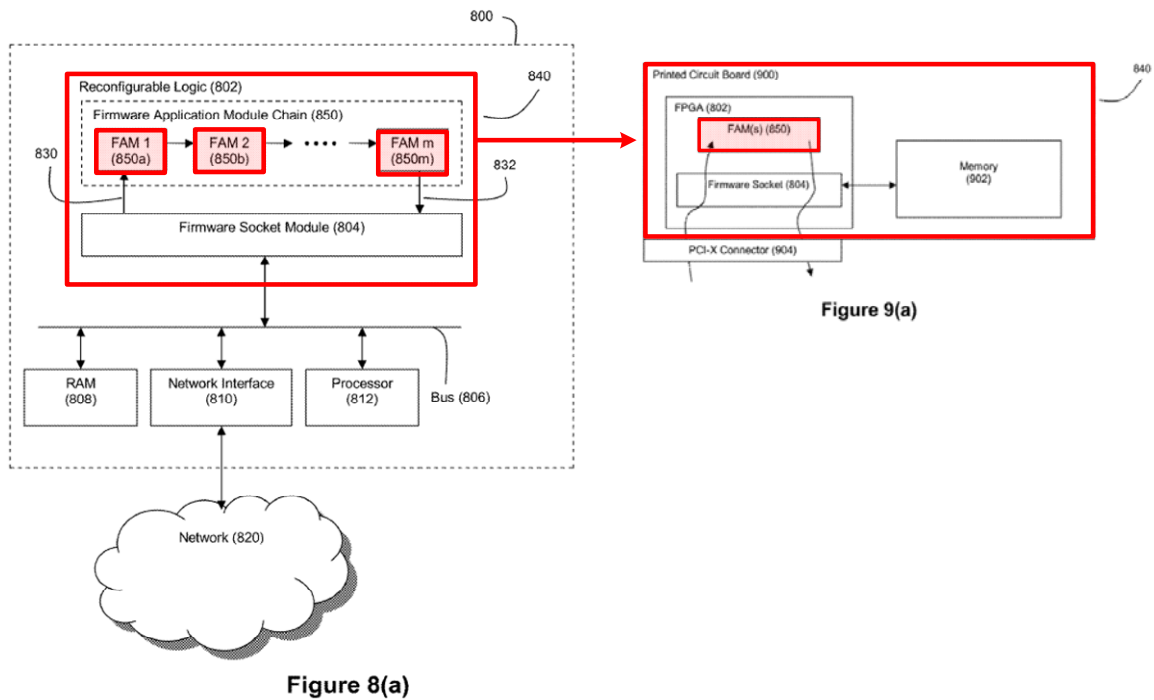
main processor. *Id.*, [0039], [0052], [0056]. These are the very same devices described in the '115 patent.

103. Indeed, Parsons discloses the same FPGA-based coprocessor implemented using the same FAM pipeline configured to perform the same financial market data processing described in the '115 patent. Compare, for example, the FPGA coprocessor illustrated in Figures 2 and 4(a) of Parsons with Figures 8(a) and 9(a) from the '115 patent, both reproduced below, and both of which are configured FAM pipelines configured to perform the same processing of financial market data—they are almost identical.

Figures 2 and 4(a) of Parsons



Figures 8(a) and 9(a) of the '115 patent



104. Also compare Parsons, [0013]-[0014], [0039]-[0046], [0052]-[0056] with '115 patent, 5:1-14, 9:54-11:42-12:11, for other examples of Parsons's disclosure of the same FAM pipelines.

105. Fig. 6 illustrates Parsons's market platform 600 that consolidates financial data processing on reconfigurable logic devices, include device 604 and 606 highlighted in orange and red below.

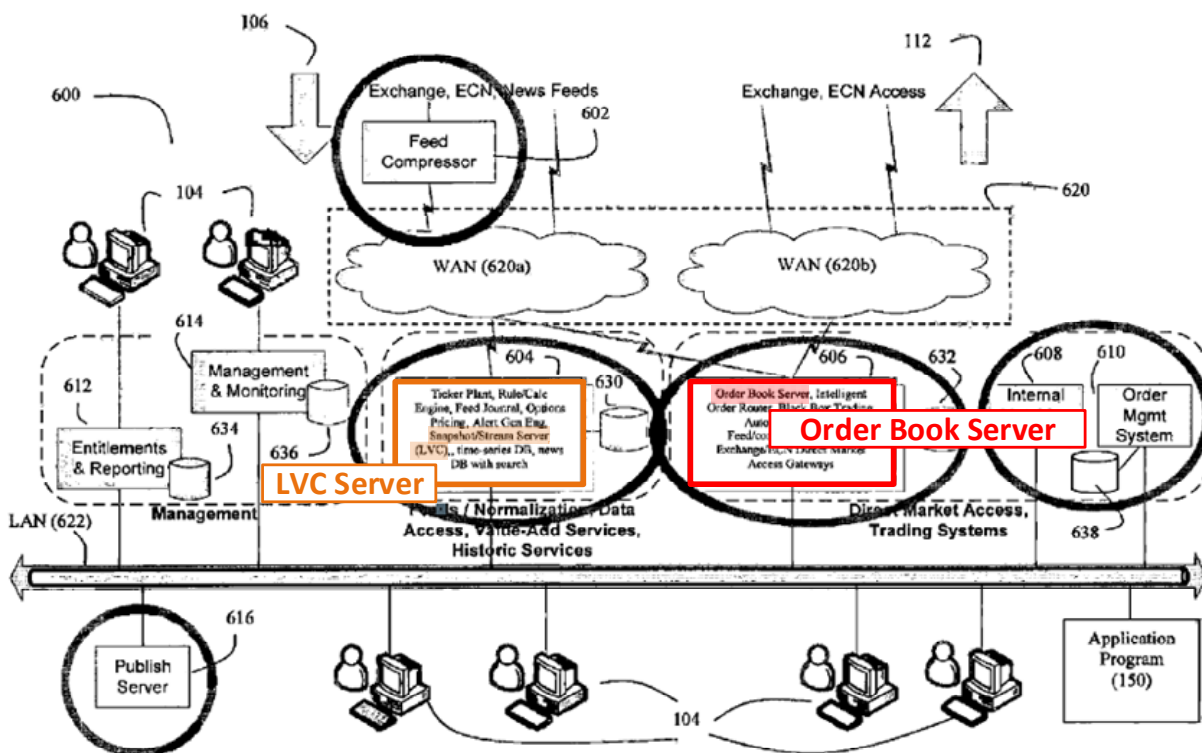


Figure 6

106. Reconfigurable logic device 604 and 604 provide a number of services including an “order book server” (OBS) and a Last Value Cache (LVC) server that process in parallel a financial market data stream 106 received from the

various exchanges and provide their respective outputs to downstream traders at workstations 104. Parsons, [0056]-[0062], [0071].

107. To perform their respective services, the OBS and LVC server employ respective caching modules configured to maintain a cache of financial instrument records in real-time as updates to those financial instruments are reported by the exchanges. *Id.*, [0060], [0062], [0071], [0095]-[0096], [0125]-[0132]. Each caching module is programmed to perform the specific updates needed to keep the data structures stored by the respective caching modules up-to-date in real-time. *Id.*

108. The LVC server uses an LVC module to a database of financial instrument records where “[e]ach record represents the current state of that financial instrument in the market place.” *Id.*, [0060], [0062]. Parsons explains that the LVC server updates the records “in real-time via a stream of update messages received from the feed handlers” and is configured to provide “real-time snapshots of financial instrument status” where “information such as the ‘latest price’ for a financial instrument can be determined.” *Id.*, [0062].

109. By maintaining the database of financial instrument records that include price point records based on the stream of update messages, the LVC server meets the “price engine” limitation of the challenged claims.

110. The ticker-plant embodiment illustrated in Fig. 11 uses the same LVC module described in the LVC server as the update FAM that maintains regional and composite price point records storing price point information reflecting the current state of financial instruments on individual exchanges (regional) and across the market (composite). *Id.*, [0062], [0126]-[0127]. The update logic that maintains these regional and composite records also meets the claimed “price engine” limitation of the challenged claims.

111. The ticker-plant embodiment illustrated in Figure 11 of Parsons is almost identical to the FAM pipeline containing the ONPA module in the ’115 patent. Compare Figure 11 of Parsons (reproduced below, left) and Figure 12(c) of the ’115 patent (reproduced below, right). The FAM modules highlighted in blue are the same, as explicitly stated in the ’115 patent. The only difference in the pipeline is that Parsons’s Figure 11 illustrates the LVC caching module, which in addition to performing many of the same operations as the ONPA, is only one of the caching modules disclosed in Parsons. Parsons, [0062] (“LVCs maintain a database of financial instrument records” that represent “the current state of that financial instrument in the market place.”), [0095], [0126]-[0127].

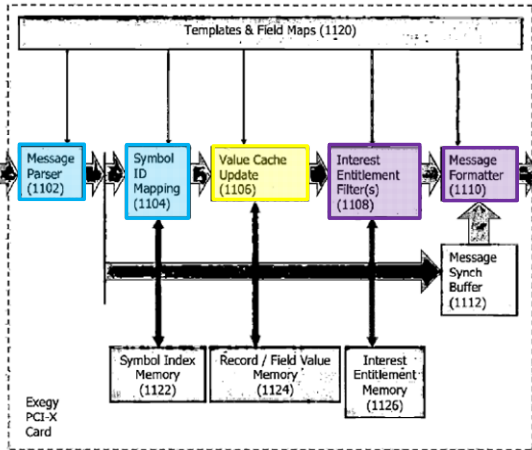


Figure 11
(Annotated)

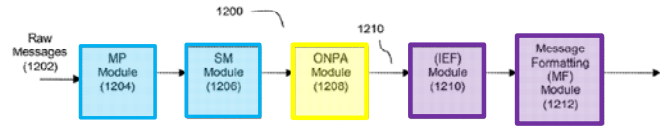


Figure 12(c)
(Annotated)

112. To maintain up-to-date order books, the Parsons's OBS employs an “order book cache update, retrieve and normalize (OBC)” module configured to update limit order records and price aggregation records for each financial instrument record stored in its cache. *Id.*, [0071], [0096]. The reconfigurable logic of the OBC configured to perform updates to the order and price-aggregated data structures meet the claimed “order engine” and “price engine” limitations of the challenged claims.

B. Parsons's Caching Modules (OBC, LVC, GVC)

113. Parsons's “caching modules” (*id.*, [0092]-[0094]) provide the “update FAM” (*id.*, [124]-[127]) in a pipeline that are configured to update a cache of financial instrument records in real time based on financial market information received from different trading venues. *Id.*, [0062], [0071], [0095], [0096].

114. Parsons discloses a generic template for a caching module, referred to as a Generic Last Value Cache (GVC), that “maintains a cache of records whose

fields are updated in real-time with information contained in streaming messages received from a message parsing module.” *Id.*, [0097].

115. To configure the GVC to perform specific operations, a programmable data dictionary is programmed to define the record structure and the update operations performed on the defined records. *Id.*, [0097] (explaining that the “structure of a record and the fields contained within it,” as well as the “type of update performed for an individual field in a record” are “defined by a programmable data dictionary.”).

116. As discussed in the previous section, Parsons discloses two specific examples of caching modules that have been programmed via the data dictionary to perform specific operations on a stream of financial market data messages to maintain and update a cache of financial instrument records: 1) a Last Value Cache (LVC) module (*see e.g., id.*, [0095]); and 2) an “order book cache update, retrieve and normalize (OBC) (*see e.g., id.*, [0096]).

C. Parsons’s Order Book Server

117. Parsons’s discloses implementing its “order book server” (OBS) on a reconfigurable logic device 606. *Id.*, [0056], [0069]-[0071]. Device 606 is an FPGA (402) on printed circuit board 400 having FAM pipeline (230) loaded thereon, as shown in annotated Figure 4(a) below. *Id.*, [0045], [0052]-[0053], [0069]-[0071].

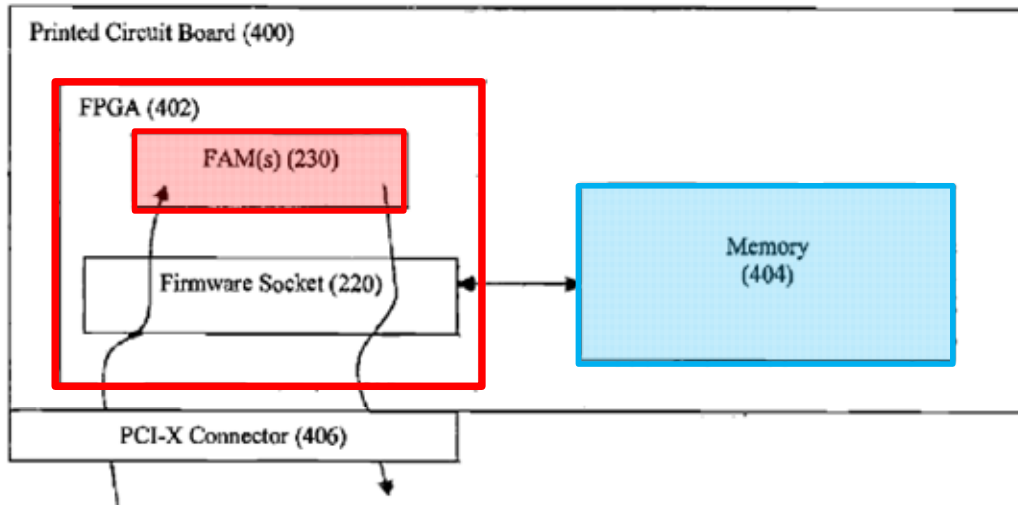


Figure 4(a)

118. The OBS maintains “financial instrument records” in memory 404 (highlighted in blue above) that include “a sorted list of the bids and offers associated with all outstanding orders for that instrument.” *Id.*, [0071]. These financial instrument records are kept up-to-date in “real-time via update messages” received from the market feeds that include “order information for each instrument [] received from a variety of different trading venues in stream 106.” *Id.* To access and update the order books in real-time, the OBS FAM pipeline implements the OBC as the update caching module. *Id.*, [0006], [0071], [0096].

119. The OBC maintains financial instrument records for each financial instrument for which the OBC maintains an order book. To support different view of the order books, each financial instrument record stores information on outstanding orders in two data structures: 1) limit order sub-records storing “a

sorted list of the bids and offers associated with all outstanding orders” for each financial instrument; and 2) price-aggregated sub-records storing “entries that are indicative of the total number of orders available at each price point.” *Id.*, [0071], [0096]. The order and price-aggregated data structures are referred to as “sub-records” because the OBC maintains them as part of the financial instrument record.

120. Specifically, Parsons discloses that each financial instrument record “consists of an array of sub-records that define individual price or order entries for that financial instrument.” *Id.*, [0096]. That is, each sub-record in the array defines either an order entry or a price entry. Those sub-records in the array that define individual order entries are the order sub-records having an entry for each outstanding limit order for that financial instrument. Those sub-records in the array storing individual price entries are the price-aggregated sub-records storing aggregated order information at each price point.

121. A POSA would have understood that the individual price entries refer to the “entries that are indicative of the total number of orders available at each price point” created to support price-aggregated views. To support this view, the financial instrument records would include a corresponding data structure that can be updated in real-time to keep the price-aggregated order information up-to-date to that real-time updates can be provided. The composite view confirms that views

of either the limit order data structure or the price-aggregated data structure over multiple exchanges can be provided. *Id.*, [0096].

122. As explained in section IV.A, a limit order specifies a specific financial instrument, price, the number of shares (or contracts) and whether the order is a bid or an ask (i.e., what “side” of the book the order is on). A POSA would have understood that sub-records that “define” individual order entries for each outstanding limit order would have included at least this information (i.e. symbol, price, number of shares and side of the book). The financial instrument to which the order pertains is captured by the fact that the order entries are sub-records of the corresponding financial instrument record. A POSA would have understood that the order sub-record would have include other identifying information like the order number, the exchange on which the order was sent, time of creation, etc., as well.

123. Parsons discloses that the OBC supports price-aggregated views “where orders at the same price point are aggregated together to create entries that are *indicative of the total number of orders* available at each price point.” A POSA would have understood this aggregated order information would include both the total number of orders (referred to as the “order count”) at each price point as well as the total number of shares (often referred to as the volume or total size) available at each price point, the latter of which is the information most relevant to

traders. Order count on its own is normally of less value, but during volatile trading, could be indicative of changing market demand dynamics.

124. The OBS is also configured to perform this functionality including “top slice” computations (determining all orders that are the best (highest bid, lowest offer) making up the NBBO) and price aggregation in which “orders at the same price point are aggregated together to create *entries* that are indicative of the total number of orders available at each *price point*,” which entries are the same as the ’115 patent’s price point records. ’115 patent, 18:9-16, 20:25-26 (referring to price point records as “price aggregation data structures.”).

125. Parsons also describes an “order book cache update, retrieve and normalize (OBC)” module (Parsons, [0096]) configured to perform the above described operations for the OBS. Specifically, the OBC maintains a cache of financial instrument records, each having an array of sub-records defining the outstanding orders for that financial instrument sorted by price. *Id.*, [0096]. This sort ordered list of orders for each instrument provides real-time updates to traders subscribing to the different views supported by the OBC (*id.*, [0096]) of outstanding orders for these financial instruments.

126. The OBC updates sub-records in real-time. *Id.*, [0006], [0071], [0096] (“sub-records are updated in real-time with information contained in streaming messages ... Sub-records associated with a record are created and

removed in real-time according to information extracted from the message stream”).

1. The OBS FAM Pipeline

127. Parsons does not illustrate in the drawings a FAM pipeline specifically for implementing the OBS, but a POSA would have understood that Parsons’s OBS implemented on reconfigurable logic device 606 (*id.*, [0069]-[0071]) would have been implemented using the appropriate FAM modules disclosed, including Parsons’s OBC caching module specifically programmed to perform the financial instrument record and message stream update functionality described for the OBS. *Id.*, [0071], [0077], [0084], [0096].

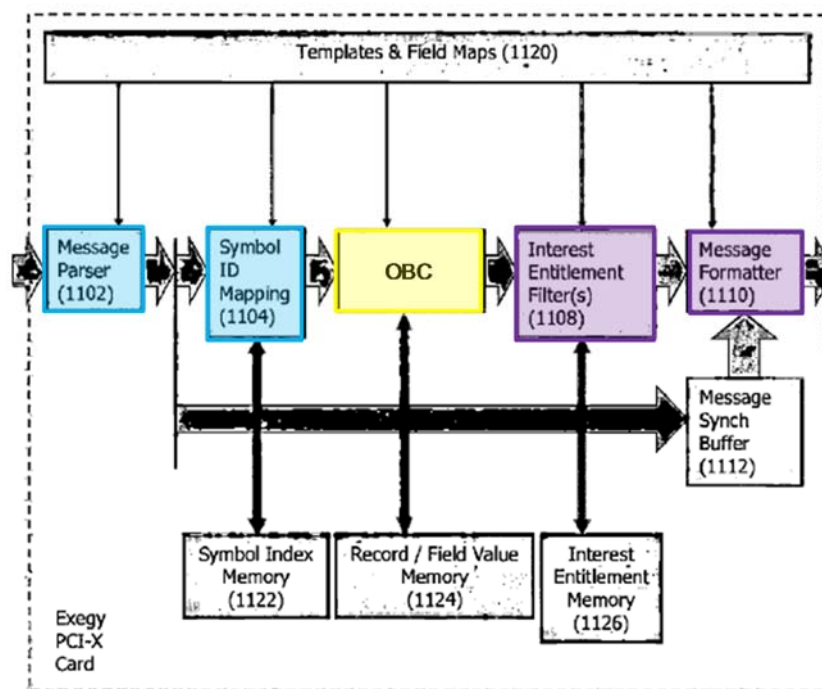
128. Parsons’s OBS utilizes the OBC caching module described as performing the OBS functionality in conjunction with the supporting FAM modules configured to process and provide updates to a stream of financial market data. *Id.*, [0077] (“appropriate FAM modules and corresponding FAM pipelines to implement these various functions for device 606 can be carried out by a person having ordinary skill in the art.”).

129. Parsons illustrates a number of exemplary FAM pipelines, including “an exemplary FAM pipeline for performing the functions of an exemplary ticker plant embodiment” in Fig. 11. The pipeline in Fig. 11 includes each of the FAM modules that Parsons explicitly discloses using in conjunction with the OBC to

implement an order book server (*id.*, [0092]-[0093], [0096], [0113]), but with the price summary LVC employed as the caching module in this example.

130. An example OBS FAM pipeline implementing Parsons's OBS is illustrated below in Figure 5 using the FAM modules Parsons teaches using in connection with the OBS and OBC.

Figure 5: Exemplary OBS FAM Pipeline Implementing Parsons's OBS



131. Specifically, Parsons discloses using the above illustrated FAM modules in connection with the OBC to perform order book update and server functionality. Specifically, Parsons is explicit that the OBC operates on the message stream from the message parsing module (e.g., Message Parser 1102 illustrated in Fig. 11). *Id.*, [0096] (the OBC updates financial instrument records

“in real-time with information contained in streaming message received from *a message parsing module*”).

132. Additionally, Parsons discloses that symbol mapping module is used to map the financial instrument symbol in the message stream to an internal number that can be used by the OBC to directly access the corresponding financial instrument record stored in the cache, citing the exemplary FAMs in Figs. 12-14 as examples (of which Symbol ID Mapping Module 1104 is one embodiment). *Id.*, [0093], [0114].

133. Similarly, Parsons discloses using an interest and entitlement filtering module to filter the stream of message coming from the OBC according to downstream consumers who are entitled to receive specific information based on their subscription, for example, traders who have subscribed to the different views or different exchanges supported by the OBC, citing to the exemplary FAM in Fig. 16 describing Interest and Entitlement FAM1108 that passes its entitlement mask to Message Formatter 1110 to construct an output message from the updated fields of the message stream and the original input stream. *Id.*, [0092], [0108], [0137]-[0138].

134. Thus, this OBS FAM pipeline shown in Figure 5 above results from loading the specific FAM modules Parsons teaches using to implement the OBS

pipeline on reconfigurable logic device 606. *Id.*, [0068]-[0071], [0077], [0083]-[0084], [0092]-[0093], [0096], [0108], [0113], [0137]-[0138].

D. Parsons's Example LVC Memory Manager and Message/Record Updater

135. Parsons discloses an LVC Memory Manager configured to retrieve records from the LVC module based on the financial instrument identified in the message stream and load those records into record buffers for processing by a set of update engines. *Id.*, [0125], [0132]. The LVC Memory Manager also retrieves records corresponding to a financial instrument identified in the next message in the stream and loads those records into the buffer so that multiple messages can be processed in parallel, thus achieving higher message throughput. *Id.*, [0128]-[0134].

136. Parsons's LVC server also has a message/record updater that distributes update operations on the records loaded into the record buffers across a set of update engines to distribute the processing load across different update engines operating in parallel to "maintain high throughput." *Id.*, [0125], [0132].

137. In the embodiment of Fig. 11, the LVC Memory Manager retrieves composite and regional records corresponding to the financial instrument identified in a message and loads the records into corresponding record buffers, as shown in annotated Fig. 15(a) below. *Id.*, [0126]-[0127].

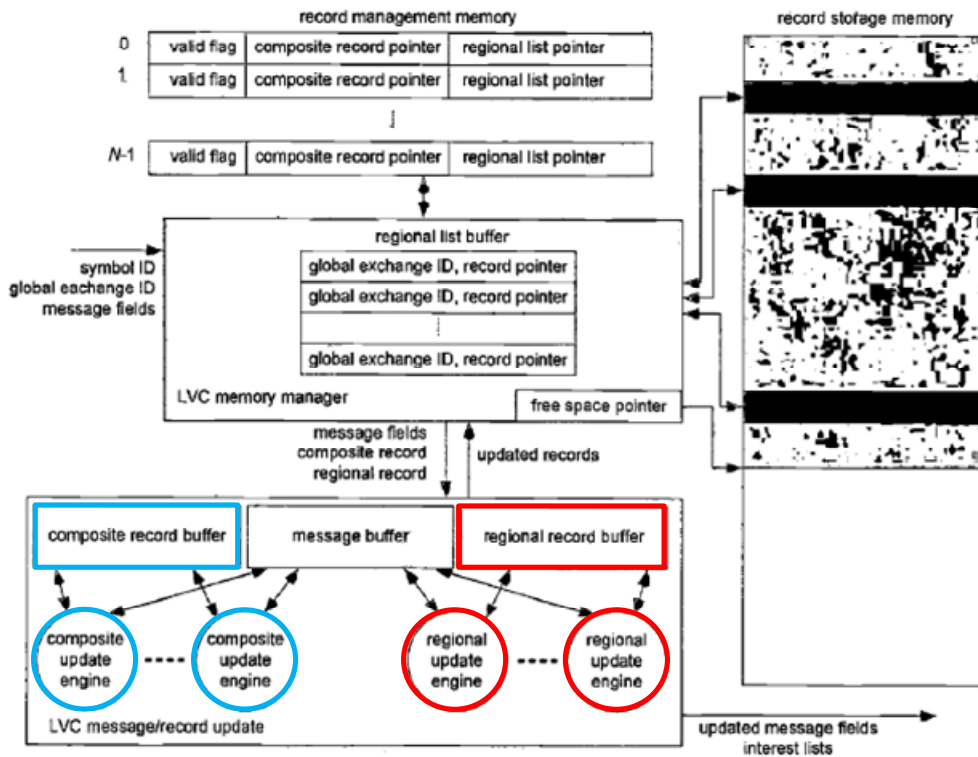


Figure 15(a)

138. The LVC message/record updater is programmed to assign update operations to different update engines configured to perform update operations for a respective record type to distribute the processing load over the set of update engines. *Id.*, [0125], [0132].

139. In this example, the message/record update assigns one set of update engines to update the composite record and a separate set of update engines to update the regional records. *Id.*, [0125], [0132], Fig. 15(a). The message/record updater manages these two sets of engines to operate “in parallel” so that accessing the record buffers and computing the calculations required to update the records,

and the writing of the updated records back to the record buffers (from where they will be written back to cache by the Memory Manager) are performed in parallel.

140. Specifically, as shown in Fig. 15(a), a first set of composite update engines configured to update composite records access and update records from the composite record buffer and a second set of regional update engines configured to update regional records access and update records from the regional record buffer in parallel. *Id.*, [0125], [0132], Fig. 15(a).

141. Thus, the LVC message/record updater is programmed to assign update operations to different update engines configured to perform update operations for a specific record type to distribute the processing load over the set of update engines. Through such parallel operation, “high throughput is maintained.” *Id.*, [0132].

VII. CLAIMS 43 AND 44 ARE OBVIOUS IN VIEW OF PARSONS

142. I have been asked to provide my opinion concerning whether claims 43 and 44 (the “challenged claims”) of the ’115 patent would have been obvious to a POSA in light of the prior art references identified in the Petition. For the reasons explained below, it is my opinion that each of the challenged claims would have been obvious to a POSA.

143. ***First***, Parsons discloses an FPGA-implemented FAM pipeline configured to operate as an order book server (OBS) to maintain and update order

books for financial instruments based on a stream of limit orders received from different exchanges. Parsons, [0071], [0096]. Parsons's OBS pipeline employs an "order book cache update, retrieve and normalize (OBC) operation" to update limit order records to maintain outstanding orders in the order books. *Id.*, [0096]. Parsons's OBC also updates price point records. *Id.*

144. Parsons's OBC functionality is implemented by a FAM in a FAM pipeline on an FPGA coprocessor. *Id.*, [0069]-[0071], [0077]. Parsons's OBC functionality that updates the limit order records (*id.*, [0071], [0096]) is an order engine as claimed, and the OBC functionality that updates the price point records (*id.*) is a price engine as claimed.

145. Parsons does not describe the OBC's updating of order and price records as being performed in parallel, and thus does not disclose that the two engines perform their respective operations in parallel as claimed. However, a POSA would have been motivated to implement such parallelism (i.e., so that Parsons's order and price engines perform their respective operations in parallel) for a host of reasons.

146. It was well-known that parallel processing increases speed and decreases latency, and Parsons teaches that it is desirable to "accelerate the speed of processing" in its system. *Id.*, Abstract. The obvious implementation of Parsons's OBC to implement the order and price engines in parallel satisfies every

limitation of claims 43-44 and renders those claims obvious. Parsons's OBC implemented using this parallel update engine optimization is one such reason that the challenged claims are obvious.

147. **Second**, Parsons also discloses a Last Value Cache (LVC) module that operates in parallel with the order book server (OBS). *Id.*, [0056], Fig. 6. Parsons's LVC module maintains and updates different price point records than the OBS does. The LVC module's data structure includes information on "total trade volume at bid, total trade volume at ask, traded value, traded value at bid, traded value at ask, and volume-weighted average price." *Id.*, [0126]. The records illustrating the volume traded at particular price points (e.g., the "bid" price point and the "ask" price point) are price point records as claimed. Thus, Parsons's LVC module includes a price engine as claimed.

148. A POSA would have had numerous reasons to modify Parsons's disclosed implementation to implement the OBS and LVC on the same FPGA, including consolidating this functionality on a single FPGA reducing hardware, reducing redundant processing by sharing common FAMS, simplifying the process of delivering this financial market data to downstream subscribers among others. This obvious modification to Parsons to consolidate the OBS reason that the challenged claims are obvious.

149. By the earliest possible priority date for the '115 patent (December 2008), FPGA resources had increased from when Parsons was filed (June 2007) and a POSA would have understood that the number of FPGAs used to implement Parson could be reduced. A POSA would have been motivated to combine the OBS and LVC functionality onto a single FPGA, and to have the OBS's order engine and the LVC's price engine continue to perform their respective functions in parallel.

A. The Challenged Claims Are Obvious Over Parsons's FPGA-Implemented Order Book Server

1. Discussion of Claim 43

150. Challenged claim 43 is broadly drawn to a coprocessor configured to receive streaming limit order events (e.g., update messages from the exchanges reporting new limit orders, modifications to existing orders, completed trades, etc.) and that has two engines configured to perform operations in parallel: 1) an order engine configured to access limit order records and compute updated limit order data; and 2) a price engine configured to access price point records and compute updated price point data. The coprocessor is also configured to enrich the stream of limit order events with financial market depth data—i.e., any data that represents the state of a financial market.

Claim 43
[43PRE] An apparatus for applying specific computer technology to reduce latency and increase throughput with respect to streaming data enrichment, the apparatus comprising:
[43A] a coprocessor that includes an order engine and a price engine;
[43B] wherein the coprocessor is configured to receive streaming data representative of a plurality of limit order events pertaining to a plurality of financial instruments;
[43C] wherein the order engine is configured to (1) access a plurality of limit order records based on the streaming limit order event data, and (2) compute updated limit order data based on the accessed limit order records and the streaming limit order event data;
[43D] wherein the price engine is configured to (1) access a plurality of price point records based on the streaming limit order event data, and (2) compute updated price point data based on the accessed price point records and the streaming limit order event data;
[43E] wherein the coprocessor is further configured to enrich the streaming limit order events with financial market depth data based on the computed updated limit order data and price point data; and
[43F] wherein the order engine and the price engine are configured to perform their respective operations in parallel with each other as the limit order event data streams through the coprocessor.

151. Limitations [43C] and [43D] recite conventional operations performed on limit order information provided via market feeds from the various exchanges as described in section IV. '115 patent, 3:38-4:67

152. The '115 patent does not purport to have invented these operations and in fact acknowledges them to have conventionally been performed by “downstream components of a market platform” (*id.*, 4:67-5:1) to construct order

books, as described in the background of the '115 patent in connection with Figs. 2(a) (illustrating an order book constructed to store each outstanding order sorted by price) and 2(b) (illustrating an order book constructed to store the number of orders and available shares available at each unique price point). *Id.*, 4:1-11.

153. For example, limitation [43C] merely recites conventional order book operations performed to maintain the “full order depth” data structure for the order book construction illustrated in Fig. 2(a) and limitation [43D] merely recites conventional order book operations performed to maintain the “price aggregated depth” data structure for the order book construction illustrated in Fig. 2(b). The '115 patent acknowledged in its background that both of these order book constructions were conventional. *Id.*, 4:1-11, Figs. 2(a), 2(b).

				order #	price	time	size		
Sell order <div>339973 \$25.42 09:51:03 250</div>				329091	\$25.43	09:47:23	50	Ask side	Top of Book
				328432	\$25.43	09:46:31	25		
				329881	\$25.42	09:51:03	100		
				329880	\$25.41	09:51:02	5		
				328128	\$25.40	09:45:31	20	Bid side	
				329127	\$25.40	09:45:31	10		
				329192	\$25.40	09:48:20	500		
				330821	\$25.39	09:59:23	200		
				329012	\$25.38	09:47:01	50		

Figure 2(a)

order count				price	time	size	
Sell order 339973 \$25.42 09:51:03 250				\$25.45	09:47:23	540	Ask side
				\$25.43	09:46:31	75	
				\$25.42	09:51:03	100	
				\$25.41	09:51:02	5	
				Top of Book			
				\$25.40	09:45:31	530	Bid side
				\$25.39	09:59:23	200	
				\$25.38	09:47:01	50	
				\$25.36	09:45:31	1500	
				\$25.35	09:48:20	1240	

Figure 2(b)

154. What the '115 patent purports to have invented is moving these conventional operations to construct and maintain order books to a “ticker plant” employing “a coprocessor that serves as an offload engine to accelerate the

building of order books” (*id.*, 5:1-14). This is, precisely the solution proposed years before by Parsons using the same reconfigurable logic device to perform the same functions (Parsons, [0006]-[0015], [0071], [0096]).

155. Specifically, Parsons’s order book server maintains order books having both the data structures illustrated above to support providing real-time updates to support different views of the order books. *Id.*, [0071], [0096].

156. Limitation [43E] recites the enriching function for which these reconfigurable logic devices were designed to perform (*id.*, [0011], [0013], [0056], [0062], [0069]-[0071], [0095]-[0096], [0126]), and limitation [43F] recites a performance optimization technique also disclosed by Parsons (*id.*, [0125], [0132]).

157. Limitation [43F] recites the obvious way to implement Parsons’s order and price engines to perform their functions in parallel to maintain high throughput of its FPGA coprocessors that are designed perform time-sensitive computations in real-time at hardware speeds. *Id.*, [0010]-[0015], [0096], [0132], [0162]. It would have been obvious to implement Parsons’s OBC to use the parallel updates engines described in Parsons to maintain high throughput through the pipeline, as discussed below. *Id.*, [0125], [0132].

2. A POSA Would Have Been Motivated to Implement the OBC with LVC Memory Manager and Record/Message Updater

158. The OBC performs cache accesses based on a record key or record ID computed from the financial instrument identified in the message stream like the other caching modules. *Id.*, [0093]. Parsons does not describe the details with respect to the interface between the update logic and the memory, or describe details of how the cache is accessed. A POSA would have been had reasons to use the Memory Manager that Parsons teaches for the LVC, modified to retrieve records stored according to the OBC's record structure, so that multiple messages could be processed in parallel to "achieve higher message throughput." *Id.*, [0134].

159. The OBC and LVC are both caching modules that maintain and update financial instrument records stored in a cache. Also, like the LVC, the OBC performs updates on two different record types in maintaining its order books. *Id.*, [0071], [0096]. A POSA would have been motivated to use the disclosed LVC message/record updater modified to assign update operations performed on limit order sub-records and price-aggregated sub-records to different update engines operating in parallel to distribute the processing load so that "a high-throughput is maintained." *Id.*, [0132].

160. A POSA implementing parallel update engines to update the OBC's financial instrument records to increase throughput would have modified Parsons's OBC ("Modified-OBC") by assigning order sub-record updates and price-aggregated sub-record updates to different update engines operating in parallel. Distributing update operation across different update engines based on the type of record being updated is the same scheme describing for the LVC maintaining composite and regional records.

161. Specifically, like the LVC, for a single received limit order message, the OBC often performs updates on two different data structure records that the OBC maintains in its order books. *Id.*, [0071], [0096]. In particular, the LVC in the ticker plant embodiment illustrated in connection with Fig. 11 maintains composite and regional records and the OBC maintains order and price-aggregated records.

162. A POSA would have been motivated to implement the OBC message/record updater in the manner the updater is implemented in the LVC so that after retrieved order and price-aggregated records are loaded into corresponding record buffers, different sets of update engines are assigned to update the respective record types so that accesses to the respective record buffers and computations to update the respective records and the message are performed in parallel. *Id.*, [0125], [0132].

163. For example, a set of update engines would be assigned to the order and price-aggregated sub-records respectively so that the following operations are performed in parallel with each other: (1) access the corresponding record buffer, and (2) perform the computations to update the respective sub-records so that the Memory Manager may write the updated up-records back to cache.

164. Performing these operations in parallel in the OBC for the order and price-aggregated sub-records would distribute the processing load and increase processing speed so that “a high-throughput is maintained.” *Id.*, [0132]. A POSA would have been motivated to do so given the “latency sensitive” nature of the OBC’s real-time operations. *Id.*, [0071], [0096]; *see also id.*, [0010]-[0015], [0071], [0096], [0162].

165. More specifically, Parsons teaches that when different record types are to be maintained and updated (e.g., the order sub-records and the price-aggregated sub-records), one way to distribute the processing is to assign a separate engine (or group of engines) the task of updating a particular record, so that records of different types are updated by different engines. *Id.*, [0125], [0132], Fig. 15(a).

166. Thus, Parsons discloses assigning update operations to different update engines to perform operations appropriate for respective records. *Id.*, [0132]. This distribution of operations across different update engines based on

record type is how Parsons describes distributing update operations in the LVC update FAM (where different engines update the composite and regional records) as illustrated in annotated Fig. 15(a) below.

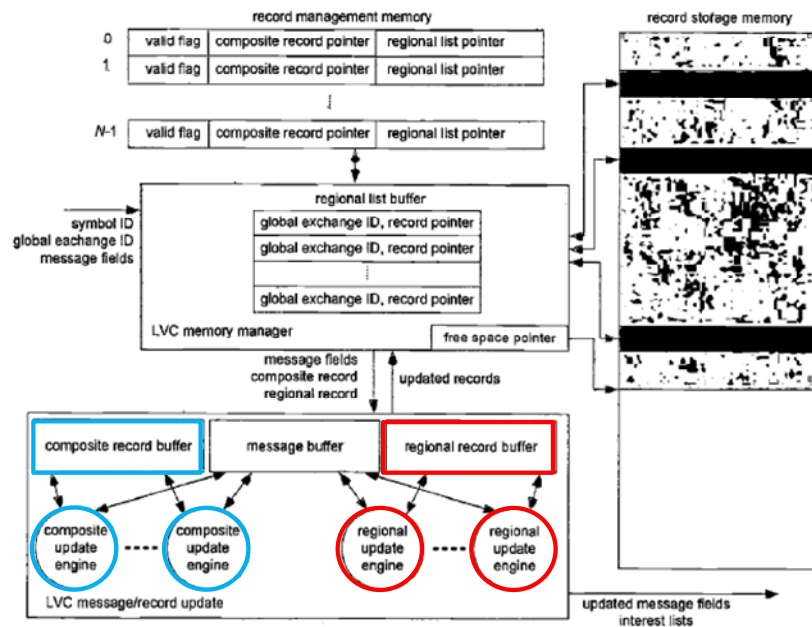


Figure 15(a)

167. The LVC message/record update distribution scheme assigns update operations performed on composite records to “composite” update engines that access a composite record buffer and operate in parallel with different “regional” update engines assigned to perform update operations on regional records stored in a corresponding regional record buffer. *Id.*, [0132]. Distributing update operations to the OBC’s order sub-records and price-aggregated sub-records to different respective sets of update engines operating in parallel would have merely been the

use of the known technique using parallel update engines configured to update different record types to yield the predictable result of increasing throughput.

168. Thus, for the reasons discussed above, a POSA would have been led to implement the OBC by having a record/message updater that applies the same type of distribution of operations by record type as used in the LVC and in other of Parsons's caching modules. *Id.*, [0125]-[0132], Fig. 15(a). When implemented in this manner, the OBC employs an order engine and a price engine that the record/message updater manages to operate in parallel as shown below in Figure 6 to perform the update operations Parsons describes of the OBC. In modified Fig. 15(a) above, the order and price engines each is implemented via multiple update engines that operate on different fields of the record to achieve further parallelism and throughput as Parsons teaches for the LVC's composite and regional engines. Parsons, [0132]. It also would have been obvious to use a single update engine to implement the order and/or price engine in the OBC.

Figure 6: Modified-OBC With Parallel Order and Price Engines

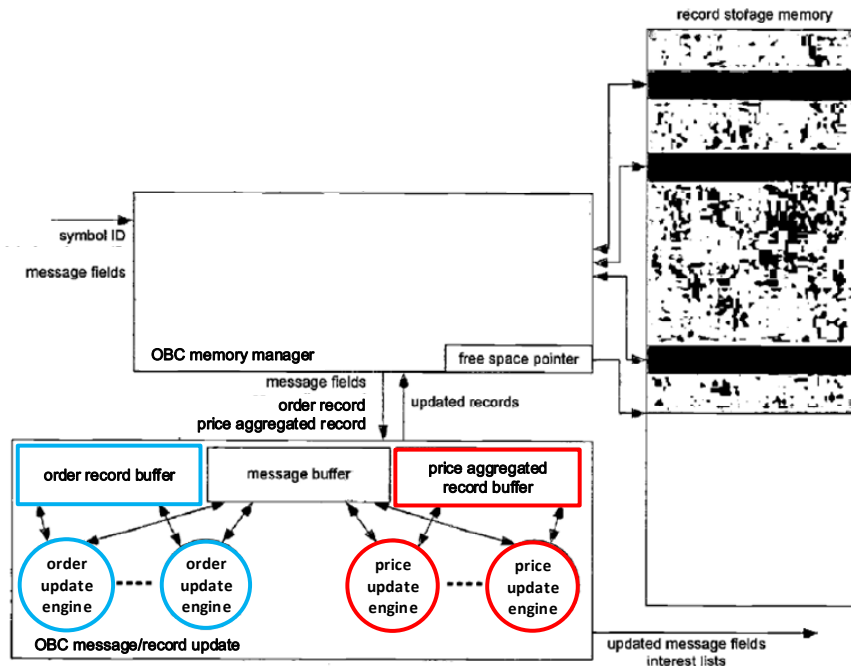


Figure 15(a)
(Modified)

169. The above-discussed implementation of the OBC, which distributes order sub-record updates across corresponding “order” update engines, and operations performed on price-aggregated sub-records are distributed across corresponding “price” update engines so that order sub-records and price-aggregated sub-records can be accessed from their corresponding record buffers and updated in parallel, is referred to herein as the “Modified-OBC.” Parsons’s Order Book Server (OBS) implemented using the Modified-OBC is referred to herein as the Modified-OBS.

170. The Modified-OBC includes all of the functionality of the OBC with the additional aspects discussed above, so that the subject matter described as being met by the OBC is also met by the Modified-OBC. Likewise, the Modified-OBS includes all of the functionality of the OBS and also meets any subject matter met by the OBS. The Modified-OBS meets every limitations of claim 43 and thus renders claim 43 obvious.

3. Claim 43

- a. **[43PRE] “An apparatus for applying specific computer technology to reduce latency and increase throughput with respect to streaming data enrichment”**

171. To the extent the preamble of claim 43 is limiting and not merely an intended use, Parsons discloses such an apparatus. Specifically, Parsons discloses using “reconfigurable logic, such as Field Programmable Gate Arrays (FPGAs)” that “can be deployed to process streaming financial information at hardware speeds” (Parsons, [0010]) to process “vast amounts of streaming financial information” at “hardware speeds via reconfigurable logic” (*id.*, [0011]).

172. Parsons discloses that, in conventional systems, “latencies are introduced” into processing of financial data “because of the processing overhead involved in transmitting messages between different machines” (*id.*, [0008]), so that consolidating this functionality on FPGAs that operate at hardware speeds reduces this latency “while providing faster data processing capabilities relative to

the conventional market data platform” (*id.*, [0008]). *See also, id.*, [0044] (“FAM operate[s] at hardware speeds (thereby providing a high throughput of target data through the FAM).

173. Thus, Parsons’s use of FPGAs on which parallel engines are deployed to process and enhance a stream of financial market data meets the preamble language because it is an application of “*specific computer technology*” that is explicitly stated as being used “*to reduce latency and increase throughput with respect to streaming data enrichment.*”

174. As shown in Fig. 2 of Parsons, this FPGA-implemented device (i.e., “coprocessor” per below) can be employed in an exemplary system 200 (i.e., “*an apparatus*”) including “the computer system’s main processor,” which is an “*apparatus for applying specific computer technology to reduce latency and increase throughput with respect to streaming data enrichment*” that comprises the claimed coprocessor, which is the ***only*** element recited for the apparatus claimed in claim 43.

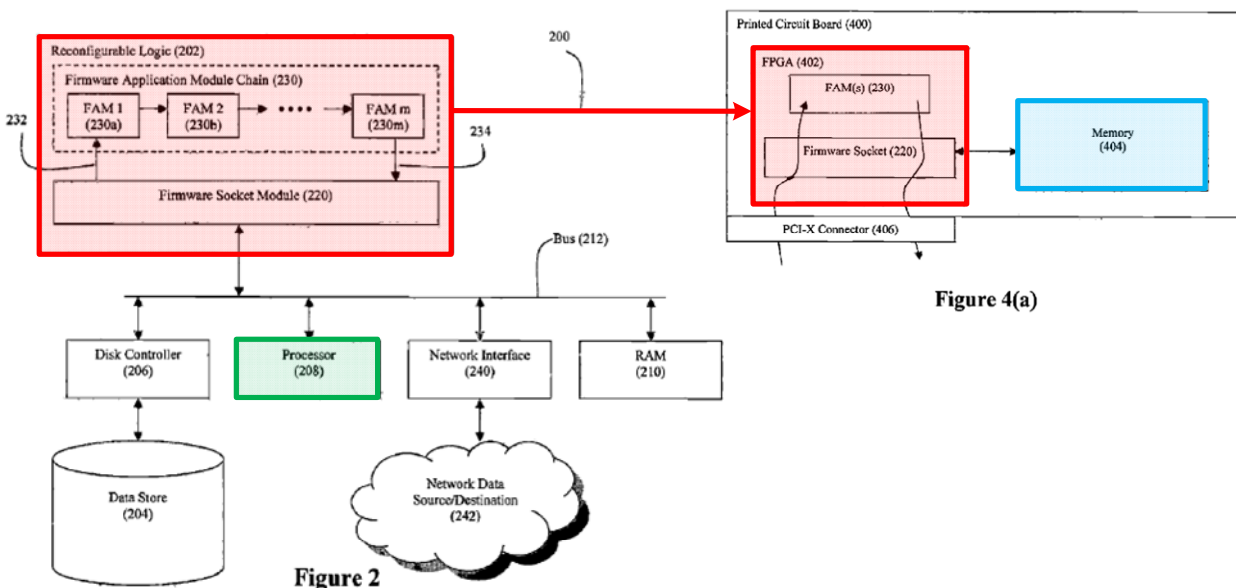
b. [43A] “a coprocessor that includes an order engine and a price engine”

175. Limitation [43A] has three parts: 1) a coprocessor, where that coprocessor includes; 2) an order engine; and 3) a price engine. Parsons’s OBS

(and thus the Modified-OBS) implemented on a reconfigurable FPGA device meets all three.

i. “a coprocessor...”

176. Parsons’s discloses implementing an OBS on reconfigurable a logic device 606 using the reconfigurable logic 202 of system 200 implemented on board 400, as shown in annotated Figure 2 and 4(a) below. Parsons’s Modified-OBS can be implemented on an FPGA in the same way.



177. As shown, this reconfigurable logic device 202 is “positioned to receive data that streams off either or both a disk subsystem defined by disk controller 206 and data store 204” (*id.*, [0039]) and is connected to “the computer system’s main processor 208” and “the computer system’s RAM 210” (*id.*) and is therefore a “coprocessor” according to the definition provided by the ’115 patent. Compare ’115 patent, 2: 58-67 with Parsons, [0056] (disclosed market platform

600 “*offloads* much of the data processing performed by the GPPs of the functional units 102 to reconfigurable logic.” In addition, the reconfigurable logic device is connected to other system components and therefore “operates in conjunction with other components in a computational system have a main processor,” as defined in the ’115 patent.

178. Reconfigurable logic device 202 is therefore a “coprocessor” according to the definition in the ’115 patent because its “a computational engine designed to operate in conjunction with other components in a computational system having a main processor.” ’115 patent, 2:58-67. While Parsons does not use the term “coprocessor” to describe the disclosed reconfigurable FPGA devices, the ’673 provisional incorporated by reference into the ’115 patent (’115 patent, 1:12-16) in fact *does*. Ex. 1003, 37 (“An exemplary platform in which the disclosed pipelines may be deployed is the *coprocessor* architecture disclosed in the 2008/0243675 publication”—*i.e.*, Parsons).

179. Thus, Parsons’s Modified-OBS performing the disclosed functionality and implemented on a reconfigurable FPGA device configured to communicate with a system’s main processor meets the claimed “*coprocessor*.”

ii. “an order engine and a price engine”

180. The ’115 patent uses the term “engine” to describe a wide range of computational elements, from the coprocessor to other elements of varying size,

complexity and function, e.g., “exchange matching engines,” “basket calculation engines,” “compression engines,” “sorting engines,” “hash engine,” etc.

’115 patent, 2:58-594:5-6, 9:45, 13:13-14, 18:27, 35:65. Thus, “engine” in the patent is set of computational elements that perform a particular function. Thus, a POSA would have understood the claimed “order engine” and “price engine” to be computational elements that perform the respective functions recited for them in claim 43. Neither “order engine” nor “price engine” is defined in the ’115, nor a term of art. *Id.* Thus, a POSA would have understood the claimed “order engine” and “price engine” to be computational elements that perform the respective functions recited for them in claim 43

181. In the ’115 patent, both the order engine and the price engine are disclosed as part of the *same* ONPA module and are characterized by what data structures they respectively maintain. *Id.*, 20:25-32. (“In the preferred embodiment, parallel engines update and maintain the order and price aggregation data structures parallel”). Parsons’s OBC also updates and maintains order and price aggregation data structures and therefore includes an order engine and a price engine for the same reasons.

182. Specifically, the financial instrument records maintained by Parsons’s OBC store the order books for each financial instrument in an array of sub-records that include order entries for each outstanding order and price entries storing

aggregated order information at each price point. Parsons, [0071], [0096]. This array of sub-records is updated in real-time based on order information in the message stream from the different trading venue to keep the order books up-to-date. *Id.*

183. To maintain the book for each financial instrument, Parsons's OBC employs two data structures: 1) an order data structure comprising an array of sub-records that define individual order entries for that financial instrument sorted by price; and 2) a price aggregated data structure comprising entries storing the total number of available orders at each price point. *Id.*, [0071], [0096].

184. Specifically, the order data structure comprises a price sorted "array of sub-records that define individual price or order entries for that financial instrument" (*id.*, [0071]). This price-ordered data structure for each outstanding order corresponds to the example conventional order book construction illustrated in Fig. 2(a) (below) referenced in the background of the '115 patent as traditionally maintained by downstream components ('115 patent, 4:1-7, 63-67).

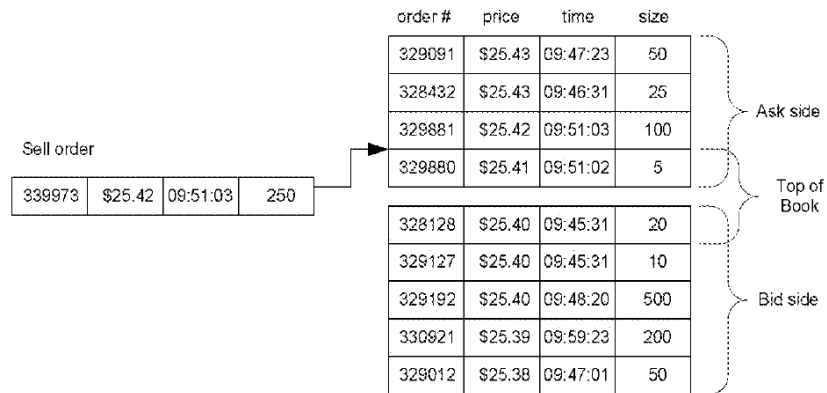


Figure 2(a)

185. The reconfigurable logic of Parsons’s OBC that updates this order data structure (Parsons ([0071], [0096])—i.e., the claimed “*plurality of limit order records*”—meets each of the limitations of the claimed “*order engine*” recited in limitation [43C], as discussed further below in section VII.A.3.d.

186. Similarly, Parsons’s price-aggregated data structure—i.e., “entries that are indicative of the total number of orders at each price point” (Parsons, [0071]) corresponds to the other example conventional order book construction illustrated in Fig. 2(b) (below) of the ’115 patent’s background storing aggregated orders (total order count and share size) at each unique price point.

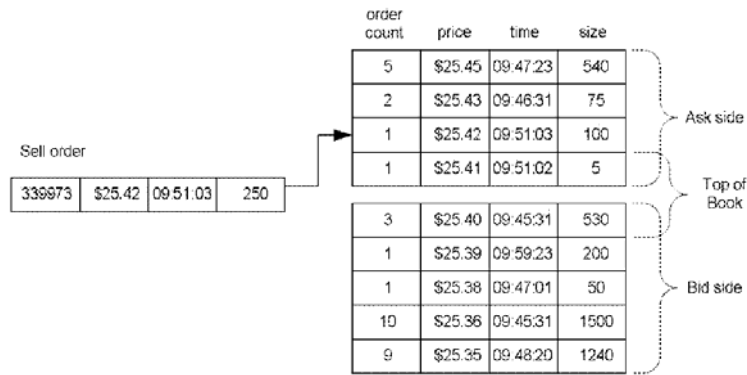


Figure 2(b)

187. The reconfigurable logic of Parsons’s OBC that updates this price aggregated data structure (Parsons ([0071], [0096])—i.e., the claimed “*plurality of price point records*”—meets each of the limitations of the “*price engine*” recited in limitation [43D], as discussed in detail in section VII.A.3.e.

188. Parsons’s reconfigurable logic device implementing Parsons’s OBS (i.e., the claimed “coprocessor”) meets [43A] because the OBC is the caching module in the FAM pipeline loaded thereon (*see* section VI.A, above) and so includes the OBC’s “order engine” and “price engine,” which also perform their respective operations in parallel, as discussed in section VII.A.3.g, below, in connection with limitation [43F].

189. This mapping is consistent with the ’115 patent that maps the “order engine” and the “price engine” to the engines that update and maintain the order and price aggregation data structures, respectively—in parallel in the preferred embodiment. ’115 patent, 20:25-30.

190. The Modified-OBC also meets limitation [43A] because it assigns order sub-record updates and price-aggregated sub-record updates to different update engines operating in parallel so that the operations recited in [43C] (as discussed further below in section VII.A.3.d) and [43D] (as discussed in detail in section VII.A.3.e) are also performed in parallel and thus meets limitation [43F], as discussed in section VII.A.3.g, below.

c. [43B] “wherein the coprocessor is configured to receive streaming data representative of a plurality of limit order events pertaining to a plurality of financial instruments”

191. Like all of the reconfigurable logic devices disclosed in Parsons, the input source is “financial data stream 106” (also referred to as “feed source” or “feed stream 106,” or simply “stream 106,” Parsons, [0006], [0040], [0057]-[0060], [0071]) received from the various financial market exchanges being tracked (also referred to as trading venues).

192. Parsons discloses that this financial data stream 106 “comprises a series of messages that individually represent *a new offer to buy or sell a financial instrument*, an indication of a completed *sale of a financial instrument*, ... notifications of corrections, and the like,” each of which reports on an event regarding a limit order. *Id.*, [0006].

193. These market events reported by the exchanges (e.g., “new offer[s] to buy or sell a financial instrument,” “a completed sale of a financial instrument,” etc.)—referred to as “order add, modify and delete events” in the ’115 patent (’115 patent, 3:50-62)—are referred to as “quote,” “trade,” and “cancel” events in Parsons ([0125]). Stream 106 therefore includes “*streaming data representative of a plurality of limit order events pertaining to a plurality of financial instruments.*”

194. Parsons’s reconfigurable logic device implementing Parsons’s OBS (i.e., the claimed “*coprocessor*”) maintains its order books in real-time by processing “update messages” that include “order information for each instrument [] received from a variety of different trading venues in stream 106.” Parsons, [0070]-[0071], [0096].

195. In particular, Parsons’s OBS maintains “a sorted list of *bids* and *offers* associated with all *outstanding orders*” that is updated based on the “order information for each financial instrument” received from the different trading venues via stream 106. *Id.*, [0071]. These outstanding orders are *limit* orders (as opposed to market orders) because a “bid” specifies the maximum price a buyer is willing to pay for a specified number of shares, and an “offer” (also referred to as an “ask”) specifies the minimum price at which a seller is willing to sell a specified number of shares—i.e., the definition of a limit order. ’115 patent, 3:55-59 (“As used herein, a ‘limit order’ refers to an offer to buy or sell a specified number of

shares of a given financial instrument at a specified price.”). Parsons does not use the term “limit order” but a POSA would have understood Parsons’s bids and offers to comprise limit orders. Parsons maintains order books with orders sorted by *price* and price-aggregated records of orders for a plurality of financial instruments at various *prices*. These are limit orders.

196. A POSA would have understood that this “order information” is the same financial market data stream from the different trading venues also includes the new offers to buy or sell financial instruments, trades of financial instruments, etc.—i.e., the quote, trade and cancel events reported by the various exchanges. Parsons, [0006], [0071], [0096], [0125]. Thus, the reconfigurable logic implementing Parsons’s OBS is configured to “*receive streaming data representative of a plurality of limit order events pertaining to a plurality of financial instruments.*” *Id.*, [0071], [0096]. Thus, the Modified-OBS meets limitation [43B] because it receives the “order information” in stream 106 that includes “*a plurality of limit order events pertaining to a plurality of financial instruments.*”

197. Parsons’s OBC is configured to update order books based on “streaming messages” from a parsing module. *Id.*, [0096]. Parsons describes a number of parsing modules that receive stream 106 and parse the stream into its constituent messages. *Id.*, [0103]-[0105], [0113] (Message Parser FAM 1102

ingests a stream of messages, parses each message into its constituent fields, and propagates the field to downstream FAMs.”).

198. Thus, the message stream from the parsing module provided as the “streaming messages” (*id.*, [0096]) to Parsons’s OBC is the parsed input stream 106 that includes the order information so that the message stream process by the OBC also includes the claimed “*streaming data representative of a plurality of limit order events pertaining to a plurality of financial instruments.*” *Id.*, [0112], Fig. 11.

- d. **[43C] “wherein the order engine is configured to [43C.1] (1) access a plurality of limit order records based on the streaming limit order event data, [43C.2] (2) compute updated limit order data based on the accessed limit order records and the streaming limit order event data”**

199. The Modified-OBC meets limitation [43C] because Parsons’s OBC updates a cache of financial instruments records having an array of sub-records storing all outstanding orders for each financial instrument based on the limit order events (“order information”) received from the different trading venues. *Id.*, [0006], [0071], [0096].

200. Limitation [43C.1] recites “the streaming limit order event data,” which refers to the “streaming data representative of a plurality of limit order events” that is recited in [43B]. Parsons’s Modified-OBS pipeline processes this

order information using a parsing module that parses the event stream into its constituent messages and provides them as a “message stream” to the OBC to update the order books. *Id.*, [0096], [0103]-[0105], [0113] (“Message Parser FAM 1102 ingests a stream of messages, parses each message into its constituent fields, and propagates the field to downstream FAMs.”). Thus, the message stream on which the Modified-OBC’s update operations are performed includes the “the streaming limit order event data” (as claimed) parsed into limit order event messages. *Id.*, [0071], [0096].

201. As discussed above, Parsons’s Modified-OBC maintains a cache of records for each of a plurality of financial instruments storing a “sorted list of the *bids* and *offers* associated with *all outstanding orders* for that instrument,” which are limit orders.

202. Additionally, a POSA would have understood that “outstanding orders” refer to limit orders since shares in a market order are typically traded immediately at the best bid or ask price on the market, and are therefore not maintained in an order book of “outstanding orders” (the impact of market order trades are however reflected in the order book by updating the number of shares in the limit order against which the market order trade was transacted).

203. To store this sorted list of limit orders, Parsons’s OBC maintains a cache of financial instrument record that each include a price-sorted array of sub-

records that define the individual order entry. *Id.*, [0096]. Parsons's OBS is configured to update financial instrument records in real-time based on the "update messages" (*id.*, [0071]) received from the trading venues reporting on new limit orders and trades (i.e., the "order information" (*id.*) from the exchanges reporting on trade, quote and cancel events). *Id.*, [0006], [0071], [0096], [0125].

204. A POSA would have understood Parsons to be describing the known limit order data structure like the conventional "full order depth" data structure of the conventional order book construction illustrated in Fig. 2(a) of the '115 patent (below) and described in its background, which likewise stores an array of records having order entries for the bids and offers for each outstanding offer for that financial instrument sorted by price.

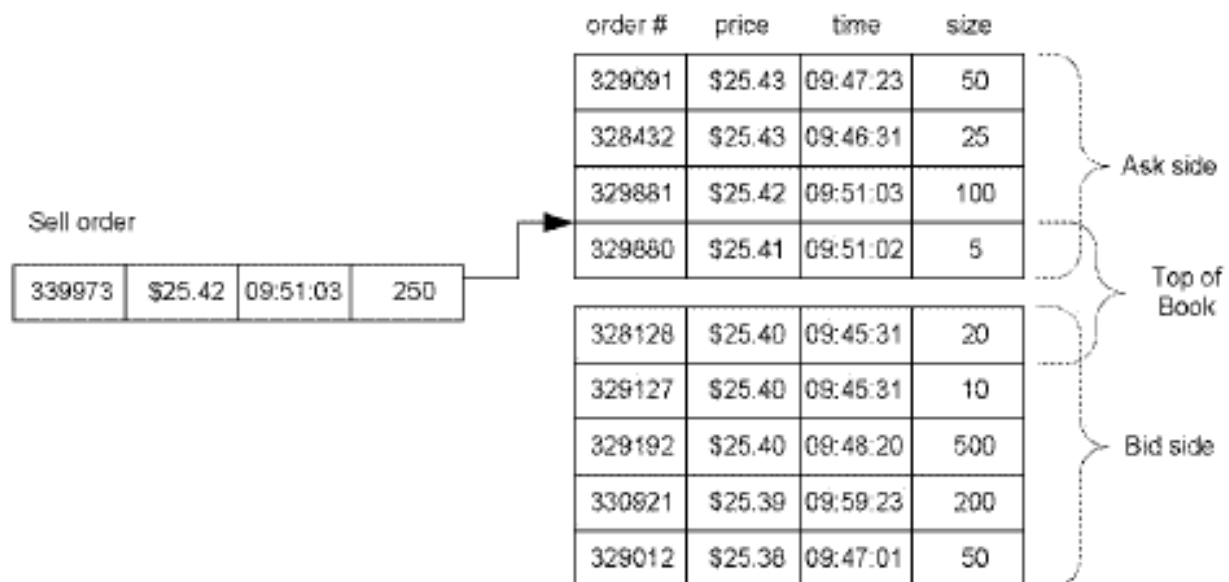


Figure 2(a)

205. Claim 43 does not require any particular field or fields to be contained in the “*plurality of limit order records*,” and the ’115 patent makes clear that limit order records can be “configured to have *more or fewer and/or different data fields*” (’115 patent, 17:25-27) than the example limit order record pictured in Fig. 20 of the ’115 patent.

206. Thus, Parsons’s array of sub-records that “define” the individual order entries for all outstanding orders (bids and offers sorted by price) meets “a plurality of limit order records.” Parsons, [0071], [0096]. To update “the fields of the sub-records” with “information contained in the streaming messages,” the sub-records to be updated are first accessed. *Id.*

207. Parsons discloses that the OBC uses a direct record key number mapped from the symbol for each financial instrument to directly address the corresponding financial instrument record in the cache. *Id.*, [0093]-[0094]. The Modified-OBC retrieves order sub-records from the addressed financial instrument record and loads order sub-records into corresponding record buffers. *Id.*, [0125]-[0132]. The Modified-OBC’s update engines assigned to perform updates to order sub-records access the order sub-records stored in the record buffer to perform updates according to the information in the message stream. *Id.*, [0125]-[0131]-[0132].

208. Thus, the Modified-OBC is configured to “*access a plurality of limit order records based on the streaming limit order event data,*” because the order sub-records accessed from the record buffers are those whose fields are updated based on the order information (i.e., “*limit order event data*”) extracted from the message stream. *Id.*, [0071], [0096]. Therefore, the Modified-OBS meets limitation [43C.1].

209. As discussed above, Parsons’s OBS updates its order books in real-time in response to order information indicating new limit orders and trades (e.g., quote, trade and cancel events) received from the different trading venues via stream 106. *Id.*, [0006], [0071], [0096].

210. The operations involved in keeping order books up-to-date based on update messages from the exchanges were well-known. Ex. 1009 (Korhammer), 9:27-49 (describing updates in response to new order, change order and delete order events.). The ’115 patent acknowledges this in its background. ’115 patent, 3:50-4:67. Specifically, new limit order records are added as new limit order events are reported (and records removed when an order is fully satisfied), and existing limit order records are modified to reflect trades on a limit order for less than the full order size. Korhammer, 9:27-49. This is the minimal set of operations that would be performed to update order books.

211. Similarly, Parsons discloses update operations performed by the OBC to update the cache of financial instrument records in response to each event type. Performing any one of these operations meets the claimed “*compute updated limit order data*.” Parsons, [0006], [0071], [0096] (“The fields of the sub-records are *updated* in real-time *with information contained in streaming messages* ... Sub-records associated with a record are *created* and *removed* in real-time according to information extracted from the message stream”).

212. The update operations performed by Parsons’s OBC (and thus the Modified-OBC) meet [43C.2] at least because they are the same operations performed by the ’115 patent’s ONPA module in response to add, modify and delete events so that computing “*updated limit order data based on the accessed limit order records and the streaming limit order event data*” covers Parsons’s order book updates to the same extent it covers those same operations in the ’115 patent. ’115 patent, 17:52-18:5.

213. Specifically, the ’115 patent states that “[o]nce the record is located, the ONPA module updates fields in the record and copies fields from the record to the message, filling in fields as necessary to normalize the output message...the ONPA module may modify the type of the message type. *Id.*, 17:52-18:5. None of these operations are described as computing “*updated limit order data*,” a term that is nowhere used in the specification (the term “*limit order data*” is used to

generically describe data contained in the input stream, *see e.g., id.*, 4:41, 6:51-7:26, 9:62, 12:11-34). Each of these operations are performed by the Modified-OBC. Parsons, [0096], [0100], [0125].

214. The only specific update described in the '115 patent is in response to a trade of shares for less than the full order. '115 patent, 17:64-18:5. In this “scenario, if the outstanding order was for 150 shares, the ONPA module would ***update the size field 2014 of the limit order record*** to replace the 150 value with 50 reflect [sic] the removal of 100 shares from the order” (*id.*). The Modified-OBC also updates the fields of the sub-records in real-time based on trade events in the message stream. Parsons, [0071], [0096], [0125].

215. A POSA would have understood that to maintain order books for “all outstanding orders” (*id.*, [0071]) as new orders and trades are streamed from the exchanges (*id.*, [0006], [0071]), Parsons’s update operations (“the fields of the sub-records are updated in real-time with information contained in streaming messages” (*id.*, [0096])) would have included updating these fields to accurately reflect available shares in an outstanding order in response to a trade event for less than all of the shares in the order’s bid or ask. *Id.*, [0071], [0096].

216. Specifically, A POSA would have understood that Parsons’ updating would have included computing a new quantity value for an outstanding limit order in response to a trade on that order so that the corresponding record

accurately reflects the current state of the outstanding order on the market—and in any event, doing so would have been obvious to a POSA as a conventional approach consistent with the types of computations Parsons discloses offloading to reconfigurable logic. *Id.*, [0011]-[0015], [0056], [0071] (OBC is “configurable to perform updates by “***computing a new value*** based upon ***message and/or record or sub-record fields*** as guided by a programmable formula” by programming its data dictionary), [0096].

217. Otherwise, the order books for each financial instrument maintained by Parsons’s OBS would not accurately reflect the state of “all outstanding orders for that financial instrument (*id.*, [0071])” for which Parsons’s OBC is configured to update in real-time.

218. Parsons’s disclosure of the same types of operations performed on a retrieved limit order record disclosed in the ’115 patent (all of which are the conventional operations performed to maintain order books) reinforces that the Modified-OBC meets limitation [43C.2].

219. As discussed above, the ’115 patent does not use the term “*updated limit order data*,” (the term “*limit order data*” is used to generically describe data contained in the input stream, ’115 patent, 4:41, 6:51-7:26, 9:62, 12:11-34). The ’115 patent merely describes a number of generic operations performed by its ONPA module in connection with a located limit order record. *Id.*, 17:52-18:5

(“ONPA module updates fields in the record and copies fields from the record to the message, filling in missing fields as necessary to normalize the output message).

220. Claim 43 is therefore not limited to any particular type of computation nor is the claimed “*updated limit order data*” limited in any way, so that creating a new sub-record with limit order data extracted from a new order event in the message stream (Parsons, [0096]) also meets limitation [43C.2], as this operation involves multiple computations to extract the limit order data from the “update message” (*id.*, [0071]) in the message stream and populate the fields of the newly created order sub-record (i.e., “moving the data field from the message to the sub-record,” (*id.*)) with the limit order data of the new limit ordered reported. *Id.*, [0096].

221. This new sub-record *is* “*updated limit order data*” that is computed to update the accessed financial instrument record based on the limit order event data in the message stream, so that Parsons’s OBC also meets limitation [43C.1] when it creates a new order sub-record for the accessed financial instrument record to store the new limit order reported by the exchange. *Id.*, [0071], [0096].

222. As discussed above, one additional example of how limitation [43C.2] is met, a POSA would have understood the Modified-OBC computes a new quantity value for an outstanding limit order in response to a trade that partially

(but not fully) fulfills that order, so the updated record accurately reflects the current state of the outstanding order. *Id.*, [0011]-[0015], [0056], [0071] (OBC is “configurable to perform updates by “computing a new value based upon message and/or record or sub-record fields as guided by a programmable formula” by programming its data dictionary), [0096]. To the extent Parsons is not considered to disclose this, that would have been the conventional and obvious way to update an order book of the type Parsons describes.

223. Thus, the Modified-OBS meets [43C.2] for any of these update operations performed by the Modified-OBC.

- e. **[43D] “wherein the price engine is configured to [43D.1] (1) access a plurality of price point records based on the streaming limit order event data, and [43D.2] (2) compute updated price point based on the accessed price point records and the streaming limit order event data”**

224. Parsons’s Modified-OBS performs price aggregation where “orders at the same price point are aggregated together to created entries that are indicative of the total number of orders available at each price point.” Parsons, [0071]. The ’115 patent describes its price aggregation functionality in almost identical terms as Parsons. For example, *compare* ’115 patent, 18:6-16: “the OPNA module may additionally perform price aggregation in order to support price aggregated views of the order book,” *with* Parsons, [0071], describing that the OBS is configured to

present the order book in different views including: “a price aggregate view where orders at the same price point are aggregated together to create entries that are indicative of the total number of orders available at each price points”), and Parsons, [0096] “The OBC includes the ability to generate various views of the book for a financial instrument including but not limited to a price-aggregated view.”

225. As discussed above, each financial instrument record in the cache maintained by Parsons’s OBC stores an array of sub-records having an entry for each outstanding order for that financial instrument. *Id.*, [0071], [0096]. The array of sub-records for each financial instrument record also store the price-aggregated entries for that financial instrument. Specifically, Parsons discloses that each financial instrument record “consists of an array of sub-records that define individual price or order entries for that financial instrument.” *Id.*, [0096]. That is, each sub-record in the array stores either an order entry or a price entry. A POSA would have understood that the price entries correspond to the aggregated price point “entries that are indicative of the total number of orders available at each price point.”

226. Parsons’s price-aggregated entries store the *same* information as the ONPA’s example price point records (also referred to as “price aggregation data structures,” ’115 patent, 20:25-26)—i.e., the number of outstanding orders at each

price point. *Id.*, 18:6-16 (“[in] this data structure, a record is maintained for each unique price point in an order book” and “preferably” contains “the price, volume (sum or order sizes at that price []), and order count (total number of order at that price).”).

227. A POSA would have understood Parsons to be describing the known price-aggregated data structure like the conventional “price-aggregated depth” data structure of the conventional order book construction illustrated in Figure 2(b) of the ’115 patent (below) and described in its background, which likewise stores an array of records having price-aggregated entries for aggregated orders at each price point sort by price.

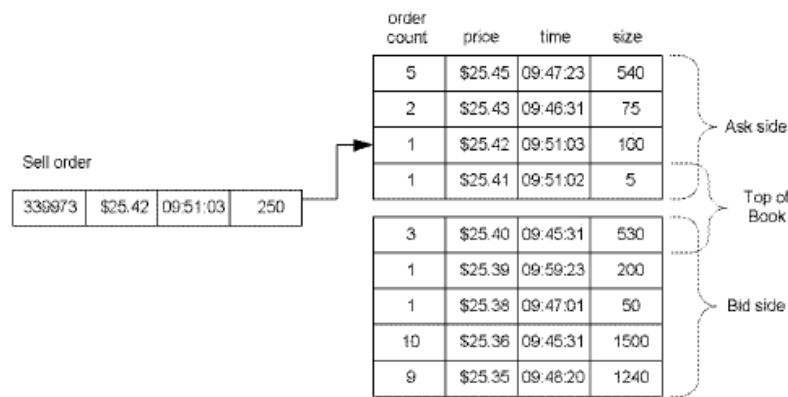
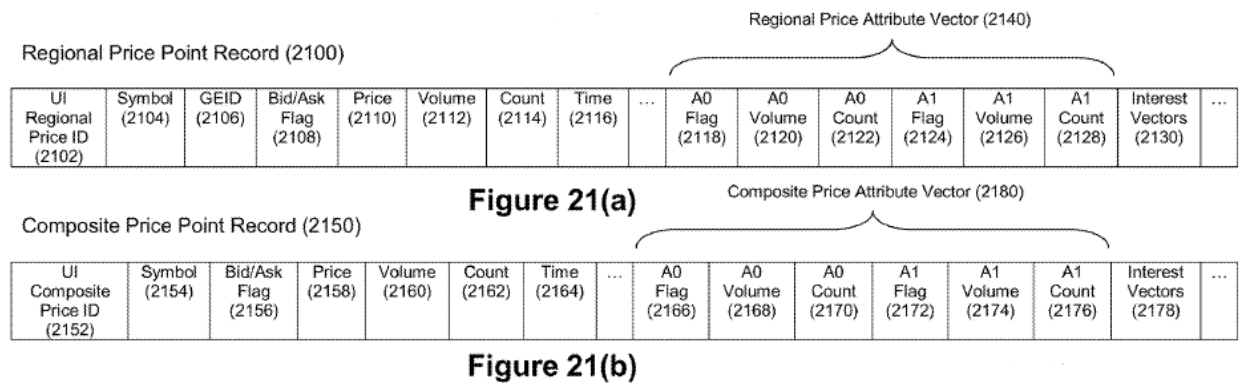


Figure 2(b)

228. At minimum, Parsons’s “entries that are indicative of the *total number of orders* available at each *price point*” (Parsons, [0071]) include the “Price” and “Count” fields in the exemplary price point records in Figs. 21(a) and 21(b) of the ’115 patent. Claim 43 does not require the claimed price point records

to include any particular field and the '115 patent makes clear that price point records can “*have more or fewer and/or different data fields.*” '115 patent, 20:22-24. Thus, the price-aggregated sub-records having price entries for each price point meet the claimed “*plurality of price point records.*”



229. Additionally, a POSA would have understood that entries “*indicative* of the total number of orders available at each price point” would have included the aggregated number of shares in those orders and not strictly the order count, as the total number of shares available at each price point is the price-aggregation information of primary interest to traders. At minimum, this would have been the obvious way to implement Parsons’s OBC to support the disclosed price-aggregated views. Thus, a POSA would have understood Parsons’s price-aggregated sub-records to store at least the same information that the '115 patent says its price-aggregated data structures “preferably contain.” '115 patent, 18:11-16.

230. As discussed above in connection with updating order sub-records, to update the fields of the price-aggregated sub-records, the sub-records are first accessed by directly addressing the corresponding financial instrument record (Parsons, [0093]-[0094]) and loading the price-aggregated sub-records to be updated into corresponding record buffers. *Id.*, [0125]-[0132]. The Modified-OBC's update engines assigned to perform updates to price aggregated sub-records access the price-aggregated sub-records from the record buffer to perform updates according to the information in the message stream. *Id.*, [0125]-[0131]-[0132].

231. Thus, the Modified-OBC is configured to “*access a plurality of price point records based on the streaming limit order event data*,” because the price-aggregated sub-records accessed in the record buffers are those whose fields are updated based on the order information (i.e., “*limit order event data*”) extracted from the message stream. *Id.*, [0071], [0096]. Therefore, the Modified-OBS meets limitation [43D.1].

232. The Modified-OBS maintains order books (including the price-aggregated sub-records) based on the messages reported from the exchanges, and the Modified-OBC does so by updating, creating and removing sub-records to keep the order books up-to-date in real-time. *Id.*, [0006], [0071], [0096]. Parsons does not describe the simple computations involved in computing updated price point data to maintain price-aggregated records but those computations were known.

However, the operations involved in keeping order books that maintain price-aggregated data structures up-to-date were also known to a POSA as discussed above.

233. Specifically, to ensure that order books accurately reflect up-to-date aggregated order information at each price point at which shares are available via the outstanding limit orders, new price-aggregate records are added when new limit orders at a new price point are reported, and existing price-aggregated records are modified to reflect new total share volume in response to new limit orders and trades of shares reported at an existing price point.

234. The Modified-OBC is configured to perform all of these types of update operations, and performing any one of them meets the claimed “compute updated price point data.” *Id.*, [0006], [0071], [0096] (“The fields of the sub-records are *updated* in real-time *with information contained in streaming messages* ... Sub-records associated with a record are *created* and *removed* in real-time according to information extracted from the message stream”).

235. For example, a POSA would have understood that updating “the fields of the sub-records...in real-time with information contained in streaming messages,” (*id.*, [0096]) would have included computing a new total share volume and/or new total order count in response to new limit order and trade events reported by the exchanges so that the corresponding financial instrument record

accurately reflects the price-aggregated order information at each available price point.

236. To the extent that Parsons is not considered to explicitly disclose this, that would have been the conventional and obvious way to maintain up-to-date price-aggregated records so that the order book reflects current information. *Id.*, [0071], [0096]. For example, to maintain the conventional price-aggregated depth structure illustrated in Fig. 2(b) of the ‘115 patent, each of the values store would have been updated to reflect changes to those records resulting from new limit orders and trades being reported from the exchanges at new and existing price-points, as discussed above. These computations would have been performed on order books in which such price-aggregated data structures were maintained, otherwise they would not reflect current financial market information.

237. Parsons’s OBC is disclosed as being configurable to perform such computations by programming the data dictionary to compute new values based on information extracted from the message stream and accessed sub-records. *Id.*, [0096]. The obvious way to implement these computations in Parsons’s order book server would have been to program the OBC (via its programmable data dictionary) to compute updated price-aggregated share volume and order count (i.e., “*updated price point data*”) by summing the number of shares in a new limit order message in the message stream at an existing price point with the share

volume/size field and incrementing the order count field in the accessed price-aggregation entry at that price point (i.e., “*based on the accessed price point records and the streaming limit order event data*”). *Id.*, [0096].

238. Similarly, the obvious way to implement Parsons’s OBC would have been to also program the OBC to subtract the number of shares in a trade event at an existing price point reported in the message stream from the share volume/size field in the accessed entry, and if the trade is for all the shares in an order, decrementing the order count. *Id.*, [0096].

239. Thus, the Modified-OBC (and therefore the Modified-OBS) computing any of these updated price-aggregated values (i.e., “*updated price point data*”) based on new order and/or trade events extracted from the message stream and the accessed price-aggregated sub-record (i.e., “*based on the accessed price point records and the streaming limit order event data*”) meets limitation [43D.2].

- f. **[43E] “wherein the coprocessor is further configured to enrich the streaming limit order events with financial market depth data based on the computed updated limit order data and price point data”**

240. The ’115 patent describes that streaming limit order events are “enriched” by modifying fields in, or appending fields to, the input message stream with updated information pertaining to the corresponding limit order events.

'115 patent, 18:22-26; 18:61-63. Parsons discloses this same enriching of the message stream.

241. The purpose of Parsons's market platform, including the order book server, is to enrich the financial data stream received from the different trading venues and provide the enriched financial market data to downstream traders at hardware speeds. Parson, [0013]-[0014], [0056], [0069]-[0071], [0096], [0162] (explaining that using platform 600, a trader "can obtain a variety of information on the financial markets with less latency").

242. Specifically, Parsons's order book server enriches the received streaming limit order events with financial market depth data because the OBC provides real-time updates on each financial instrument's order book to support composite views (i.e., "a sort order of the price or order entries for a financial instrument across multiple exchanges"), provides real-time updates for price aggregated views (i.e., total number of shares and orders at each price point), top-slice views, and can also synthesize "top-of-book" quote streams, etc. *Id.*, [0071], [0096].

243. Parsons explains that each of its caching modules (including the OBC and LVC) that maintains financial instrument records has the ability to provide a user with a view of the data at a particular instance in time, and to augment limit order event messages in the message stream with updates. Parsons describes a

number of ways the message stream can be updated (described below), including adding message fields and modifying message fields such as from a message/record updater that Parsons describes in the LVC and that a POSA also would have been led to use in implementing the OBC. *Id.*, [0060], [0062], [0071], [0096], [0100], [0125].

244. Parsons's teaches using "programmatic field generation" operations with the OBC to "augment messages received from a message parsing module with additional fields" based on a formula that can reference other fields in the message or any record maintained by the caching module. *Id.*, [0100]. *See also id.*, [0096] on computing new values.

245. Parsons's also teaches using multiple parallel update engines configured to perform specific update operation to message fields in the message stream according to programmed business logic (*id.*, [0125], [0132]), as discussed in section VII.g, below. Parsons discloses using a Message Formatter FAM to provide an output stream to downstream subscribers by applying updates to the message stream performed by upstream FAMs (e.g., the OBC) to the original message stream by replacing fields from the original message stream with the updated fields and/or appending fields from the updated messages to the original message stream. *Id.*, [0108]. Thus, to provide real-time updates to downstream providers, updates made by Parsons's OBC to the messages streamed through the

pipeline can be applied to the original message stream by the Message Formatter FAM by replacing fields/appending fields with the updates from the OBC so that the enriched data stream from the OBC can be put in a format expected by downstream subscribers. *Id.*, [0108], [0138].

246. As an example, a trader may seek a current view of the financial market data for a particular a particular financial instrument. *Id.*, [0071], [0096], [0160]. A current view can quickly become outdated as market conditions change for that financial instrument, which is why speed is of the essence for traders. *Id.*, [0004]-[0005]. A user requesting views of the order book for a particular financial instrument receives a refresh image of the order book corresponding to the requested view for that instrument and thereafter receives real-time updates via the output stream provided by the OBS. *Id.*, [0071], [0096], [0160].

247. A POSA would have understood that after a trader requests a view of a financial instrument, when the caching module's "message/record updater" computes an update for the data it maintains, it updates not only the cached records, but it also updates the message stream by modifying and/or augmenting the message in the message stream that triggered the update with these updates in real-time so that the trader's view of the financial market data stays current and up-to-date. *Id.*, [0005] (explaining that delays can lead to missed opportunities and

speed of information delivery can translate into significant sums of money); [0162], [0071], [0100], [0125].

248. Parsons does not describe the specific updates that the OBC performs on the message stream, but discloses that the caching modules update messages in the message stream by modifying message fields or augmenting messages to include additional fields based on other fields in the message or records maintained in the cache. *Id.*, [0100], [0125], [0132]. A POSA would have understood that the updated limit order and price-aggregated data computed by the OBC would have been applied to update the order information in the message stream to deliver these real-time updates to downstream subscribers with reduced latency. *Id.*, [0013]-[0014], [0056], [0069]-[0071], [0096], [0162] (explaining that using platform 600, a trader “can obtain a variety of information on the financial markets with less latency”). That is, a POSA would have understood Parsons to describe that the updates to the message stream are the same updates the OBC makes the order books it maintains (i.e., the cache of financial instrument records that it updates in real-time). *Id.*

249. To the extent Parsons is not considered to explicitly teach that the OBC has a message/record updater that updates messages in the message stream with the updates it computes for the order book records maintained by the OBC, the obvious way to implement Parsons’s OBS would have been to configure the

Modified-OBC to modify/augment the message stream with the order book updates that it computes so that the output stream also reflects those updates and the current state of the updated order books. Specifically, the obvious way to implement Parsons's OBS and thus the Modified-OBC to provide the disclosed updates (e.g., real-time updates to composite and price aggregation views, top-slice views, etc) is to use the techniques explicitly disclosed by Parsons to update the message stream and construct an output stream in a format expected by downstream subscribers. *Id.*, [0108], [0132], [0136]-[0138].

250. The stated purpose of Parsons's market platform is to replace all of the functional units 102 of the distributed market platform in Fig. 1 with a reduced number of hardware-accelerated appliances (e.g., FPGAs) so that this same functionality can be offloaded from the main processor and performed at hardware speeds. *Id.*, [0010]-[0015], [0056], Fig. 6. This purpose would have been frustrated were computations performed to update the order books not propagated to downstream subscribers via the output message stream; the undesirable alternatives would be for these computations to have been repeated by downstream components or additional memory accesses performed by the main processor to obtain updates from the OBS. Thus, a POSA would have implemented the Modified-OBC so that when the Modified-OBC computes an update to its order and/or price-aggregated sub-records triggered by a streaming

limit order event message, the Modified-OBC augments that streaming limit order event message with the computed update data using Parsons's disclosed message modify/augment techniques. Parsons, [0010]-[0015], [0056], [0096], [0162]. This would have enabled a downstream trader who previously requested a view of the order book to keep its order book up to date, by providing the trader with real-time, reduced-latency information on the current state of the market, which Parsons's reconfigurable logic devices are intended to do.

251. The '115 patent does not use the term "*financial market depth data*" in connection with enriching a data stream, but instead uses this term to describe the information in the *input stream* from the market exchanges ('115 patent, 5:15-31, 9:54-63), so appears to use the term broadly to describe *any* of a wide variety of market information, including information on outstanding limit orders of a financial instrument on the market (e.g., the number and price of open buy and sell orders for a financial instrument), and therefore not limited to derived values.

252. The '115 defines the similar term "financial market data" (*id.*, 1:63-2:3), and also uses "financial market data" to describe the messages processed by the coprocessor to update order books (*id.*, 9:63-10:5 ("Such incoming data [from the exchanges] preferably comprises a series of financial market data messages, the messages representing events such as limit orders relating to financial instruments"), 12:28-34 (the order book update module receives "raw messages

1202” that “comprise a stream of financial market data events, of which at least a plurality comprise limit order events.”).

253. The ‘115 patent does not indicate that it is using the term “financial market depth data” to mean something different than “financial market data.” To the extent these terms mean the same thing, the above-discussed order and/or price-aggregated sub-records the Modified-OBC enriches the stream with meet the definition of “financial market data”; they are data derived from messages representing new offers to buy/sell a financial instrument. Parsons, [0071], [0096]. ‘115, 1:63-2:3. For example, updates reflecting new limit orders, updates to the number of shares in an order available in view of a trade, updates on deleted order, updates to price-aggregated information such as total volume and order count are all derived from the “update messages” (Parsons, [0071]) in stream 106.

254. Moreover, the updated limit order data and price-aggregated data computed by Parsons’s OBC is “financial market depth data” at least because it includes the same information by which the ’115 patent “enriches” limit order events. For example, the same updated price-aggregated data (e.g., updated total share volume/order count) computed by Parsons’s OBC is described by the ’115 patent as enriching the corresponding limit order event. *Id.*, 18:61-63.

255. Additionally, the “*financial market depth data*” the ’115 patent describes as enriching the streaming order limit events is simply the updated order

and price-aggregated information computed for the limit order event in the message stream, including the same information that Parsons's OBC computes. *Id.*, 18:22-26; 18:61-63; 20:61-21:8). Thus, "*financial market depth data*," at least covers the updated order and price-aggregated information computed by the OBC discussed in connection with limitations [43C.2] and [43D.2] above.

256. Specifically, Parsons's OBS enriches the data stream with "*financial market depth data*," at least because this information (e.g., with updates to the number of shares in each individual order, and price-aggregated updates to total share size/volume and total order count at each price point per sections VII.A.3.d and VII.A.3.e, above) is the same information used to enrich the data stream in the '115 patent. '115 patent, 18:61-63 ("The ONPA module map append the volume, order count, and price attributes to events when creating enriched limit order events 1604).

257. Though not limited in this way in the '115 patent, "depth" data for a financial instrument is sometimes used to indicate the volume of shares available in the market at each price level. To the extent financial market depth data is interpreted to require an indication of volume of shares available, the price-aggregated data the Modified-OBC enriches the stream with includes volume information and would also meet this narrower meaning of financial market depth data.

258. Therefore, the Modified-OBS (i.e., the claimed “coprocessor”) is “configured to enrich the streaming limit order events with financial market depth data based on the computed updated limit order data and price point data,” because the Modified-OBC is configured to provide its real-time updates to downstream subscribers by modifying/augmenting the message from the message stream that triggered the update using the results of that update, including the computed updates discussed in connection with limitations [43C.2] and [43D.2].

- g. [43F] “wherein the order engine and the price engine are configured to perform their respective operations in parallel with each other as the limit order event data streams through the coprocessor”**

259. Limitation [43F] requires that the order engine and the price engine perform “*their respective operations*” in parallel. The ’115 patent does not define operating “in parallel,” but dependent claim 45 makes it clear that operating in parallel can include processing performed by a pipeline of engines arranged “downstream” from one another.

260. The Modified-OBC’s order engine and price engine perform their operations in parallel as required by claim 43. Parsons teaches further parallelism. The Modified-OBC distributes operations performed on order sub-records across a corresponding set of “order” update engines (i.e., the “*order engine*”) and operations performed on price-aggregated sub-records are distributed across

corresponding “price” update engines (i.e., the claimed “*price engine*,”) as illustrated in modified Figure 15(a) below using the update engine distribution scheme based on record type taught by Parsons. Parsons, [0125], [0132], Fig. 15(a).

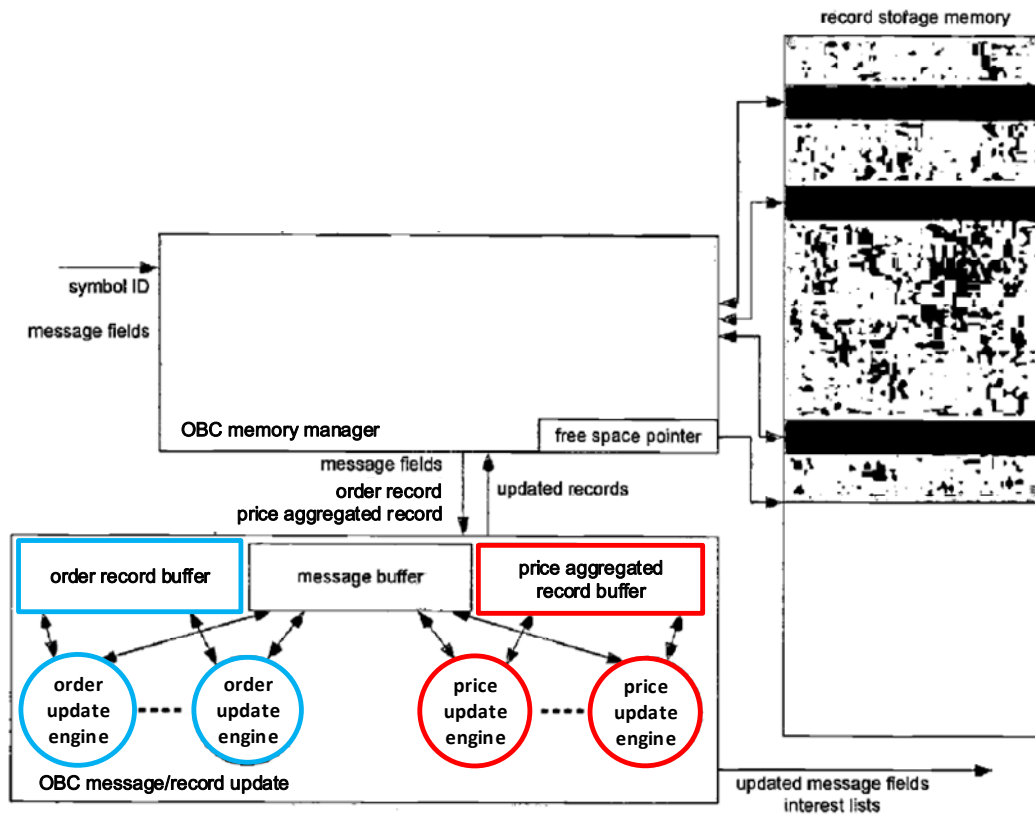


Figure 15(a)
(Modified)

261. As shown, the order sub-records and the price-aggregated sub-records retrieved from the cache for updating based on the information extracted from the message stream (*id.*, [0093]-[0094], [0096], [0128]-[0131]) are loaded into their

respective record buffers from which order sub-records and price-aggregated sub-records are *accessed* and *updated* in parallel. *Id.*

262. Claim 43 does not limit the “access” operations recited in limitation [43C.1] and [43D.1] to any particular storage or manner of accessing, so the Modified-OBC’s order engine and price engine perform their respective “*access*” operations in parallel because the order update engines assigned to perform operations on order sub-records and the price update engines assigned to perform operations on price-aggregated sub-records access their corresponding record buffers in parallel. Specifically, the Modified-OBC’s record/message updater implements the order update engines and the price update engines to access their corresponding record buffers in parallel to achieve the benefits Parsons describes by distributing update operations to different update engines configured to perform operation on the respective record types in parallel.

263. The updates to the order and price-aggregate sub-records performed by the OBC include computing the updated limit order and price-aggregated data recited in [43C.2] and [43D.2]. Thus, the Modified-OBC’s order engine and price engine perform their respective “*compute*” operations in parallel because the order update engines assigned to perform operations on order sub-records and the price update engines assigned to perform operations on price-aggregated sub-records

update their corresponding records accessed from respective record buffers in parallel.

264. Thus, the Modified-OBC's order engine and price engine perform both their respective "access" and "compute" operations in parallel and therefore meet limitation [43F].

265. I understand that if a claim does not recite sufficient structure, the claim should be interpreted as a "means-plus-function" claim under the patent statutes. The claims recite structure at least because a "coprocessor" is a known structure in the computing arts, and an "engine" is a set of computational elements that perform a particular function. I also understand that if a term is interpreted as "means-plus-function," the structure corresponding to the function needs to be described in the disclosure to be supported and described in the prior art to meet such a limitation.

266. The functions recited in claim 43 are all met by Parsons in the manner detailed above, and the structure relied upon to meet those functions in Parsons is the same (or equivalent) to the structure the '115 patent describes. Specifically, the coprocessor is an FPGA programmed with FAMS (*id.*, [0039]-[0056], Figs. 2, 4(a), 4(b)) which is the same coprocessor structure in the '115 patent. '115 patent, 9:54-12:11, Figs. 8(a), 8(b), 9(a), 9(b).

267. With respect to the order and price engines, the structure the '115 patent specification discloses for implementing each is a firmware application module (FAM) that programs the coprocessor (e.g., FPGA). *Id.*, 10:26-12:34. The order and price engines in the Modified-OBS each is implemented the same way, i.e., on FAMs that program an FPGA. Parsons, [0069]-[0096], [0100], [0125]-[0138], Figs. 2, 4(a), 4(b)), 11; *see also* section VI above. Additionally, Parsons is incorporated by reference into the '115 patent ('115 patent, 1:20-24 and 1:32-33) and thus forms a part of the supporting disclosure for the order and price engines. The '115 patent does not describe any structure that is not also disclosed in the same way or equivalently in Parsons.

4. Claim 44

268. Dependent claim 44 depends from claim 43 and merely requires the coprocessor be selected from a list of three types of coprocessors, which includes a reconfigurable logic device, of which, an FPGA is an example. Parsons's OBS is implemented on a reconfigurable logic device.

Claim 44
[44PRE] The apparatus of claim 43
[44A] wherein the coprocessor comprises a member of the group consisting of a reconfigurable logic device, a chip-multi-processor (CMP), and a graphics processing unit (GPU), and
[44B] wherein the order engine and the price engine are deployed on the member.

- a. **[44A] “comprises a member of the group consisting of a reconfigurable logic device, a chip-multi-processor (CMP), and a graphics processing unit (GPU)”**

269. Reconfigurable logic devices are the very type of coprocessor disclosed in Parsons. Parsons, [0060], [0069]-[0070], [0163]-[0165], Figures 2-6. Parsons’s OBS is implemented as a FAM pipeline loaded onto a reconfigurable logic device (*id.*, [0056], [0069]-[0071]) and therefore meets one of the “*reconfigurable logic device*” alternative recited in the claimed list of members (which I have been told is called a “Markush group” in legal jargon) and therefore satisfies this element.

- b. **[44B] “wherein the order engine and the price engine are deployed on the member.”**

270. Parsons’s OBC is loaded as the caching module FAM in the OBS pipeline (*id.*, [0056], [0069]-[0071], [0084], [0096]) and includes the order engine and the price engine and therefore both are “*deployed on the member*”—i.e., the reconfigurable logic device.

B. The Challenged Claims Are Rendered Obvious By a Reconfigurable Logic Device Implementing Both Parsons’s

OBC and price Summary LVC FAMs in the Same FAM Pipeline

1. Parsons's Market Platform

271. Parsons's market data platform (600, Fig. 6) consolidates a range of functionality (*id.*, [0014]) on fewer and smaller reconfigurable logic devices, including devices 604 and 606 on which a variety of possible Last Value Cache (LVC) server functionality and an OBS can be implemented, respectively. *Id.*, [0011]-[0015], [0056], [0060], [0070], Fig. 6.

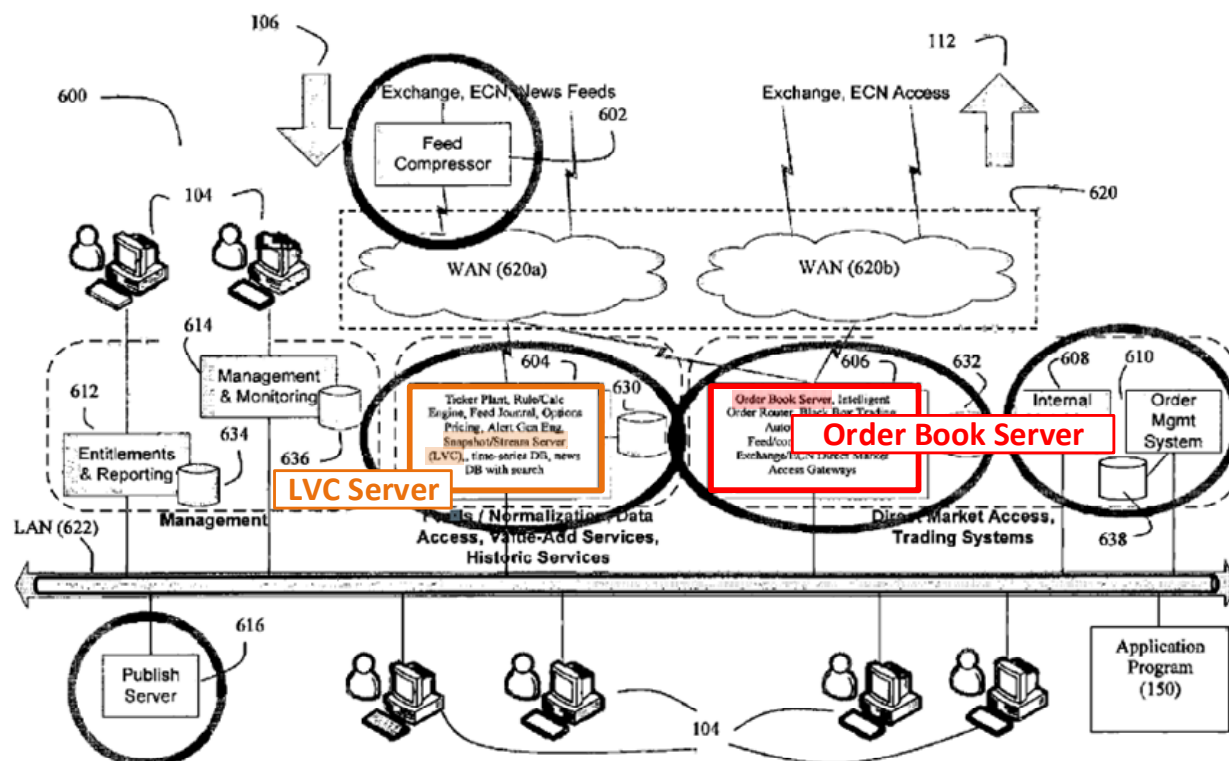


Figure 6

272. As shown in Fig. 6 of Parsons, the LVC server implemented on device 604 and the OBS server implemented on device 606 perform their operations on

financial market data 106 in parallel to deliver “a variety of information on the financial markets with less latency.” *Id.*, [0056], [0162]. Parsons explains that the system shown in Fig. 6 is merely illustrative, and that a POSA “may choose to deploy less than all the functionality described herein in reconfigurable logic,” and may arrange a device to include “some ... subset of the functions listed in Fig. 6.” *Id.*, [0165].

273. Parsons describes a number of different FAM pipelines to implement the various functionality Parsons describes, such as the pipelines illustrated in Figs. 7-11. Additionally, Parson discloses that different FAMs that can be used in different combinations and chained together in different ways to achieve desired functionality. *Id.*, [0068], [0077], [0082]. An exemplary FAM pipeline for providing “ticker plant” functionality on device 604 is illustrated in Parsons’s Fig. 11 below.

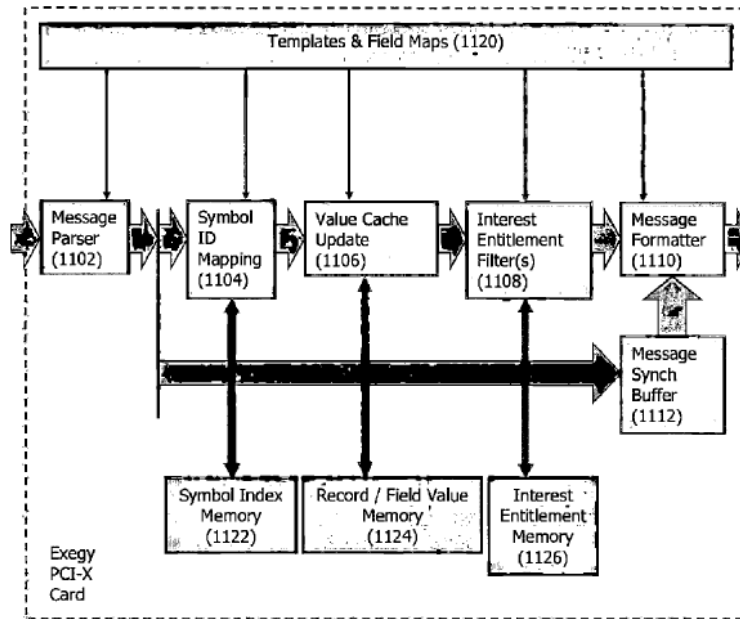


Figure 11

274. This “ticker plant” embodiment employs the price summary LVC caching module (*id.*, [0084], [0095]), illustrated as Value Cache Update (1106) and referred to in the description of the ticker plant embodiment as the Last Value Cache (LVC) Update FAM 1106 (hereinafter “LVC”). Like the OBC, the LVC performs real-time updates to financial instrument records based on a message stream from different market exchanges. *Id.*, [0062], [0071].

275. Parsons teaches that it is desirable to minimize the number of FPGAs to implement the desired functionality, and recognizes that “resource improvements over time for FPGAs” could allow for “consolidation.” *Id.*, [0011], [0056], [0165]. Indeed, Parsons even suggests that it may be possible to one day consolidate “*all* functionality shown in Fig. 6 [including the OBS and LVC] on a single system 200” with a single FPGA 202. *Id.*, [0165].

**2. A POSA Would Have Motivated to Implement Parsons's
OBS and Price Summary LVC on a Single "Ticker Plant"
FPGA**

276. Parsons discloses improving financial market data processing performance by consolidating operations conventionally distributed over a network of individual computers on fewer and smaller reconfigurable logic devices (e.g., FPGAs). *Id.*, [0006]-[0013], [0056], [0070], [0162].

277. In view of this teaching, a POSA would have been motivated to implement Parsons's OBS and LVC in the same FAM pipeline loaded onto reconfigurable logic device to take advantage of the benefits of consolidation taught by Parsons. Specifically, a POSA would have been motivated to include Parsons's OBC in the ticker plant pipeline of Fig. 11 to consolidate OBS and LVC functionality on a single device.

278. A POSA would have had reasons to consolidate this functionality on a single FPGA. For example, both the OBS and the example ticker plant process the same data stream using the same FAM modules to provide time-sensitive financial market depth data to assist traders in quickly understanding the current state of the market. *Id.*, [0062], [0071], [0095]-[0096], [0126], [0162].

279. Integrating this functionality on a single pipeline implemented on one FPGA provides numerous efficiencies, including reduced hardware (a single FPGA), reduced processing redundancy (shared FAM modules), reduced software

processing by the main processor in distributing this information to entitled subscribers, and increased simplicity for downstream components (e.g., applications and APIs), etc. *Id.*, [0092], [0136]-[0138], [0162].

280. Specifically, by providing the financial market depth data provided by the OBC and LVC in a single output data stream, less software computation (e.g., less inter-process communication, less repeated handling of the same data elements, etc.) is required by the main processor in collecting and disseminating this information to entitled subscribers, thus further reducing the latency and simplifying the process of delivering financial market depth data to subscribers interested in both types of enhancement.

281. For example, this actual view for Microsoft stock traded on INET circa 2003 combines both financial market depth information provided by Parsons's OBC (highlighted in red) and LVC (highlighted orange). *Id.*, [0071], [0096], [0126].

INET home

system stats

help

inet

MSFT

GET STOCK

MSFT

go

☐ Aggregate by Price

LAST MATCH

Price

25.8400

Time

14:06:58

TODAY'S ACTIVITY

Orders

72,283

Volume

9,463,602

BUY ORDERS

SHARES	PRICE
500	25.8400
100	25.8400
100	25.8400
600	25.8400
2,500	25.8400
92	25.8400
100	25.8400
100	25.8400
300	25.8400
200	25.8400
400	25.8400
2,000	25.8400
1,000	25.8400
500	25.8400
100	25.8400

(414 more)

SELL ORDERS

SHARES	PRICE
400	25.8500
500	25.8500
1,000	25.8500
1,000	25.8500
1,200	25.8500
600	25.8500
1,500	25.8500
500	25.8500
700	25.8500
1,500	25.8500
400	25.8500
400	25.8500
2,000	25.8500
2,000	25.8500
4,500	25.8500

(594 more)

As of 14:07:30

282. With the market platform 600 in Parsons where the OBC and LVC are implemented on separate reconfigurable logic devices, the output stream from two different devices would have to be routed to downstream consumers with subscriptions to both types of updates. *Id.*, [0133], [0136]-[0138]. By combining the OBC and LVC on the same pipeline, interest lists that specify the users that are

entitled to receive updates for both the OBS and LVC are stored on the same FPGA and applied to a single output stream constructed by the Message Formatter, simplifying the processing of the interest lists and routing of appropriate updates to entitled subscribers.

283. A POSA would have had numerous other reasons to implement Parson OBC and price summary LVC on the same device. For example, both caching modules are configured to update and provide time-sensitive financial market depth data valuable to traders in quickly understanding the current state of the market, and there are numerous efficiencies in using a single pipeline to provide this information in an integrated data stream, including reduced hardware, reduced software processing to distribute this information, etc. *Id.*, [0062], [0071], [0096], [0126].

284. Additionally, Parsons discloses that there is “tremendous value” in providing a “variety of information on the financial markets” to traders with the latency improvements provided. *Id.*, [0162]. Thus, a POSA would have been motivated to develop a single FAM pipeline to provide OBS and LVC functionality to traders, which both provide financial market depth information valuable to traders, from a single reconfigurable logic device that can be customized via the IEF and Message Formatter according to different subscriber interests and entitlements in the format expected. *Id.*, [0136]-[0138].

285. By providing the financial market depth data provided by the OBC and price summary LVC in a single output data stream, less software computation is required by the main processor in collecting and disseminating this information to entitled subscribers, thus further reducing the latency and simplifying the process of delivering financial market depth data to subscribers interested in both types of enhancement.

286. Additionally, Parsons teaches combining functionality in this very manner, explaining that some subset of the functions listed in Fig. 6 can be implemented on reconfigurable logic device 604 and a user can add further functionality by reconfiguring the device “to add any desired new functionality.” *Id.*, [0165]; *see also, id.*, [0063] (“it should be noted that numerous other FAM pipelines could be readily devised and developed by persons having ordinary skill in the art following the teachings herein”). Including the OBC in the ticker plant pipeline to reconfigure the FPGA to perform both OBS and LVC functionality would merely have been the known use of loading “hardware templates” onto an FPGA to reconfigure the device to achieve the predictable result of implementing additional functionality on the FPGA.

287. Also, Parsons contemplated consolidating OBS and LVC functionality as it envisioned FPGA improvements would allow “consolidation of *all* functionality shown in FIG. 6 on a single system 200.” *Id.*, [0165]. Indeed,

Parsons even claims a consolidated device comprising a “firmware application module pipeline” that is “configured to perform” one or more financial data processing operations from a set that includes both “a Last Value Cache (LVC) server operation” and an “order book server operation.” *Id.*, claim 24.

3. Implementing the OBS on the Ticker Plant FAM Pipeline

288. Parsons’s ticker plant embodiment processes financial market data from different exchanges and enhances the data stream with a host of price point information including last trade price, best bid price, best ask price, price direction, high trade price, low trade price, etc., and a number of derived fields such as tick direction, total trade volume, total trade volume and bid and ask, volume-weighted average price (VWAP), etc. *Id.*, [0126].

289. A POSA seeking to consolidate this price summary functionality with OBS functionality would have added the OBC “hardware template” (*id.*, [0077]) onto the pipeline as expressly taught by Parsons. *Id.*, [0165] (“to add additionally functionality to device 604, [a user] can do so by simply re-configuring the reconfigurable logic of system 200 to add any desired new functionality”).

290. The same Message Parser, Symbol Mapping, Entitlement and Filtering FAMS illustrated in Fig. 11 for the LVC are also described for use with the OBC [0093]-[0094], [0096], so that adding OBS functionality by loading the OBC into the pipeline would have required minimal and straightforward changes

to the pipeline. *Id.*, [0044] (“FAMs can also be flexibly reprogrammed to change the parameters of their data processing operations”). The Message Formatter FAM is disclosed as constructing an output stream by combining updated message streams from multiple upstream FAMs with the original message stream, and therefore would have been the obvious choice to combine the updated message streams from the OBC and LVC. *Id.*, [0108].

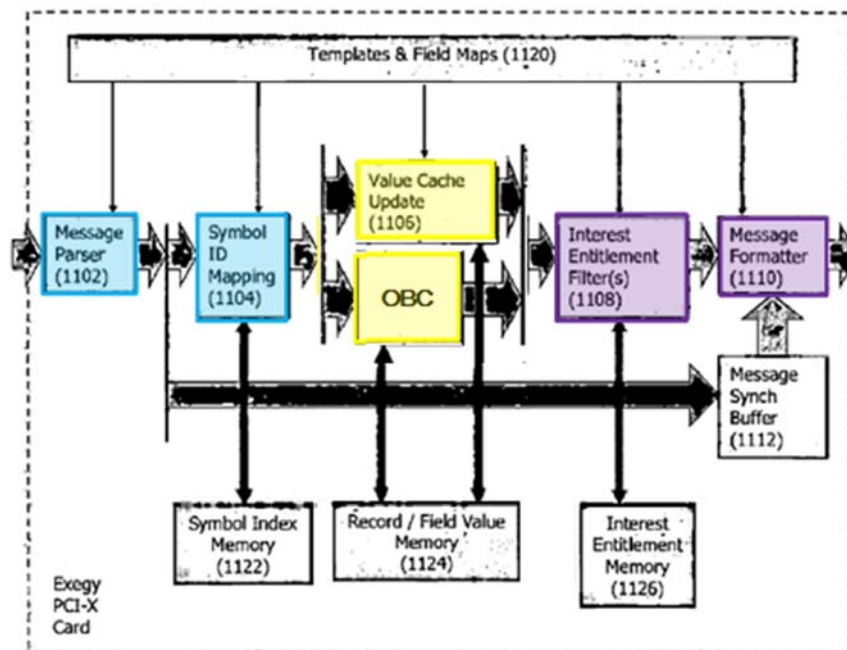
291. Thus, it would have been well within the skill of a POSA to configure the FAM pipeline to add OBS functionality to the “ticker plant” embodiment illustrated in Fig. 11 by including the OBC in the FAM pipeline and reprogramming the FAM modules to the extent needed, the simplicity of which is in fact touted by Parsons. *Id.*, [0044], [0068], [0077], [0083], [0163], [0165].

292. As discussed above, the LVC and OBS processing is performed in parallel on separate reconfigurable logic devices to deliver low latency market information to traders. Parsons, [0056], [0162], Fig. 6. A POSA would have been motivated to maintain this parallel processing when consolidating OBS and LVC functionality on a single FPGA to maintain high throughput and low latency.

293. Parsons also teaches arranging FAMs in parallel in pipelines carrying out a variety of data processing tasks. *Id.*, [0106]-[0111], Figs. 9, 10). Thus, one obvious way to implement a FAM pipeline to perform OBS and LVC functionality to maintain parallel operation is to arrange the OBC and LVC in parallel, as shown

below in Figure 7, which is the ticker plant pipeline of Fig. 11, modified to include the OBC.

Figure 7: Ticker Plant Server (“TPS”) With OBC and LVC Update Modules



**Figure 11
(Modified)**

294. This FAM pipeline loaded on a reconfigurable logic 202 (e.g., an FPGA implemented printed circuit board 400, Parsons, [0060], [0069]) is referred to herein as the Ticker Plant Server (TPS). It should be appreciated that the resource limitations mentioned in connection with consolidating *all* of the disclosed functionality onto a single FPGA (*id.*, [0165]) would not have presented an obstacle to implementing the TPS since it combines only a small subset of the functionality that was distributed across seven different devices in the disclosed

market platform (i.e., reconfigurable logic devices 604, 606, 608, 610, 612, 614 and 616). *Id.*, [0057], [0060], [0070], [0078]-[0082].

295. The TPS combines only a subset of the functionality disclosed as being implemented each of devices 604 and 606. *Id.*, [0060], [0070]. Thus, reconfigurable logic technology as of 2008 was more than sufficient to support the TPS for at least the reason it was sufficient to support the functionality disclosed as being implemented on each individual reconfigurable logic devices 604 and 606. Specifically, FPGA technology as of 2008 was sufficient to store the financial instruments records maintained by the OBC and the LVC and for the supporting FAMs in the TPS pipeline. The TPS would have needed less resources than those to support the suite of functionality implemented on device 604, for example, or the suite of services disclosed as implemented on device 606.

4. Claim 43

296. Several of the limitations are met by an FPGA implementing the TPS for the same reasons an FPGA implementing the OBS meets those limitations. Thus, for several limitations below cross reference is made back to the analysis in above for the FPGA implementing an OBS, and the cross-referenced analysis is expressly incorporated into the analysis below.

297. An FPGA implementing the TPS meets all of the limitations of challenged claim 43.

- a. **[43PRE] “An apparatus for applying specific computer technology to reduce latency and increase throughput with respect to streaming data enrichment”**

298. If the preamble is deemed limiting it is met by a Parsons FPGA implementing the TPS (hereafter “the TPS coprocessor” or the “TPS FPGA”) for much the same reasons the preamble is met by an FPGA with the Modified-OBS as discussed above in VII.A.3.a (an FPGA is “*specific computer technology*” that reduces latency and increases throughput with respect to streaming data enrichment). *See also* VII.A.3.f in the discussion below demonstrating how the TPS coprocessor enriches the streaming data.

- b. **[43A] “a coprocessor that includes an order engine and a price engine”**

299. As discussed in section VII.B.3, the TPS is implemented on a reconfigurable logic device using reconfigurable logic 202 implemented on board 400 (Parsons, [0060], [0069]) and therefore is a “*coprocessor*” for the same reasons discussed in section VII.A.3.b above.

300. The TPS coprocessor includes the computational elements of Parsons’s OBC that update the order sub-records, and those computational elements meet the claimed “order engine” because they meet each of the limitations in [43C], as discussed VII.B.4.d below, and are implemented (by a FAM on the FPGA) in the same manner as the “order engine” described in the

'115 patent. Thus, the OBC includes an order engine for the same reasons the Modified-OBC does (as discussed above in VII.A.3.d), because the modifications made to the OBC (e.g., to have the OBC's order and price engines operate in parallel) do not impact whether the OBC's computational elements that update the order sub-records are an order engine. Specifically, the elements of the OBC that update the order sub-records meet the claimed "order engine" independent of whether they operate in parallel with the elements of the OBC that update the price-aggregated sub-records.

301. The TPS coprocessor also includes the LVC that accesses and updates a cache of financial instrument records storing price point information. The computational elements of the LVC that update the price point records meet the claimed "price engine" because they meet each of the limitations in [43D], as discussed in detail below, and are implemented (by an FAM on the FPGA) in the same manner as the "price engine" in the '115 patent.

- c. **[43B] "wherein the coprocessor is configured to receive streaming data representative of a plurality of limit order events pertaining to a plurality of financial instruments"**

302. Both Parsons's OBS pipeline and the ticker plant embodiment illustrated in Fig. 11 (using an LVC as the caching module) process the same financial data stream 106 from the exchanges reporting on limit order events

(Parsons, [0006], [0057], [0062], [0071], [0112]), and therefore the TPS coprocessor is configured to receive this same financial market data stream. Thus, the TPS coprocessor meets [43B] for the same reasons the Modified-OBS coprocessor does, as discussed in section VII.A.3.c above.

- d. **[43C] “wherein the order engine is configured to [43C.1] (1) access a plurality of limit order records based on the streaming limit order event data, [43C.2] (2) compute updated limit order data based on the accessed limit order records and the streaming limit order event data”**

303. The TPS implements Parsons’s OBC and therefore meets limitation [43C] for at least the same reasons the Modified-OBC does, as discussed above in VII.A3.d. The manner in which the Modified-OBC meets [43C] is not dependent on the modifications made to the OBC to have its order and price engines operate in parallel. Parsons’s OBC meets limitation [43C] for the same reasons discussed in section.

304. In order for the OBC to update “the fields of the sub-records” with “information contained in the streaming messages” (Parsons, [0096]) the sub-records to be updated are first accessed from cache. The OBC performs cache accesses based on a record key computed from the financial instrument (Parsons, [0093]), but Parsons provides no further detail on how the OBC interfaces with its

cache. Additionally, to create and remove order sub-records the financial instrument record would be accessed in the same manner.

305. One obvious cache interface implementation for the OBC would be to use the interface Parsons describes for other caching modules, including a Memory Manager to retrieve order sub-records from cache and load them into record buffers from which the update engine (here the OBC order engine) accesses the records. Parsons, [0125]-[0132]. Since the LVC includes the computational elements that access and update price point records, as discussed below, the TPS coprocessor does not rely on the OBC to meet the price engine and thus does not rely on modifying the OBC to have its order and price engines operate in parallel.

306. Even if it were found that using a Memory Manager and record buffers were not an obvious way to implement the OBC, the OBC in the TPS coprocessor stills meets [43C.1] because irrespective of the details of how the OBC's cache interface is implemented, Parsons is clear that the OBC accesses the sub-records it updates. Parsons, [0071], [0093], [0096]. The TPS processor meets [43C.2] for the same reasons the Modified-OBC meets [43C.2] does, as discussed above in VII.A.3.d.

307. Additionally, Parsons's OBC also meets the "*compute*" limitation in [43.C] for the further reason that the updated price aggregation information is also "*updated limit order data.*" As also discussed above, the '115 patent uses the term

“*limit order data*” to generically describe data contained in the input stream. ’115 patent, 4:41, 6:51-7:26, 9:62, 12:11-34.

308. Thus, Parsons’s OBC computing updated price aggregation information (e.g., updated total share volume/size, updated order count, etc.,) also meets “comput[ing] updated limit order data” at least because it computes multiple updated data pertaining to limit orders.

309. This updated price aggregated information is computed based on the message stream and at least the corresponding financial instrument record accessed and so limitation [43C] is met for this additional reason.

- e. **[43D] “wherein the price engine is configured to [43D.1] (1) access a plurality of price point records based on the streaming limit order event data, and [43D.2] (2) compute updated price point based on the accessed price point records and the streaming limit order event data”**

310. The ’115 patent does not define a “*price point record*,” but merely provides an example, explaining that price point records can “*have more or fewer and/or different data fields*.” ’115 patent, 20:22-24. Claim 43 does not recite *any* fields that are required so that the claimed “*plurality of price point records*” are not limited to having any specific field or type of field.

311. The OBC price engine discussed for the OBS coprocessor maintains price-aggregated records for outstanding *orders*, whereas the LVC maintains

records that include price-aggregated fields for consummated *trades*. Parsons, [0096], [0126]. Specifically, Parsons's LVC maintains a cache of records that store price-aggregated trade information for individual exchanges (regional records) and across the market (composite records). Parsons, [0062], [0126]-[0127]. These composite and regional records include price point data, including raw data on price points and other data derived therefrom. Parsons, [0126]-[0127].

312. The LVC maintains composite and regional records having fields for “total trade volume at bid, total trade volume at ask, traded value, traded value at bid, traded value at ask, and volume-weighted average price.” Parsons, [0126]. Fields storing volume traded at particular price points (e.g., “bid” and “ask” price points) are price-aggregated values so that these records meet the claimed price point records for at least this reason. Fields reflecting last trade price, best bid price, best ask price, high trade price, low trade price, and close price, etc. also reflect price points so that the records maintained by the LVC are price point records for this additional reason. Parsons, [0126]-[0127]. Thus, the LVC maintains a “*plurality of price point records*” as claimed. Parsons, [0126]-[0127]. Thus, the LVC maintains a “*plurality of price point records*” as claimed.

313. Like the OBC, the LVC receives messages in financial data stream 106 (which meets “*streaming data representative of a plurality of limit order events pertaining to a plurality of financial instruments*,” as discussed in §

VII.A.3.c) via the “streaming messages received from the message parsing module” (Parsons, [0095], [0113], Fig. 11), and those messages include the same “*streaming limit order event data*” that the OBC receives (as discussed in VII.A.3.c and VII.A.3.d) and as shown in the TPS pipeline above. *Id.*; § VII.B.3.

314. Parsons’s LVC updates fields of the composite and regional records with information extracted from the message stream (i.e., *the streaming limit order data*). Parsons, [0062], [0125]-[0127]. To update its records, the LVC accesses the composite and regional records (i.e., “*the plurality of price points records*”) for the financial instrument to which a message in the message stream pertains using the financial instrument symbol and exchange identifier (i.e., the identifier of the exchange that issued the limit order event) extracted from the message stream (i.e., *based on the streaming limit order events*) using its Memory Manager, record buffers and record/message updater, as discussed above in VI.D. Parsons, [0125]-[0138]. The price engine in the LVC (i.e., the “computational elements of the LVC that update the price point records”) accesses the plurality of price point records from the record buffers and meets limitation [43D.1].

315. Parsons’s LVC computes numerous updated price point values to update the composite and regional records that meet limitation [43D.2]. Specifically, the regional and composite records “include derived fields” including total trade volume at various bid/ask price points, traded value at bid/ask, VWAP,

etc. for which updated values are computed when an input message in the message stream reports a trade of a financial instrument. Parsons, [0126]-[0127].

“*[U]pdated price point data*” as claimed covers at least any data that the LVC computes based on the accessed price point records and the streaming limit order event data (e.g., trade events) and uses to update the value of the LVC’s price point data. Parsons, [0062], [0095], [0124]-[0127].

316. Specifically, in response to a reported trade event for a financial instrument, the LVC updates the accessed composite records and regional records and the message fields according to the specific logic programmed to update records based on the event type and financial instrument type. *Id.*, [0125]-[0127].

317. Numerous of the exemplary fields disclosed in Parsons (*id.*, [0126]) involve computing updated values that are based on the information in the message stream and the information stored in the fields of the accessed records, including *each* of the derived fields mentioned above. See also the example in Parsons describing computing a new tick direction by determining “if the prices in the message is larger or smaller than the previous price captured in the record.” Parsons, [0124].

318. Each of these computed values meets the claimed “*updated price point data*.” Thus, updated price point data covers at least updated values to any of the price point data stored in the composite and regional price point records that

are computed based on reported limit order events (e.g., trade events) in the message stream processed by the LVC. Parsons, [0062], [0095], [0124]-[0127].

319. Thus, the TPS's LVC meets [43D.2] because updated price point values are computed to update the record fields based on the accessed composite and regional records and the streaming limit order event data, and because the above-referenced computed values (e.g., total volume at bid/ask, traded value at bid/ask, VWAP, etc.) meet the claimed "*updated price point data.*"

f. [43E] "wherein the coprocessor is further configured to enrich the streaming limit order events with financial market depth data based on the computed updated limit order data and price point data"

320. The TPS meets limitation [43E] because the OBC updates the message stream with updated order and price-aggregation information and the LVC updates the message stream with price summary information, which includes (among numerous other derived price point values) updated price-aggregated trade information as discussed above. Parsons, [0060], [0062], [0095], [0127]-[0133].

321. The TPS coprocessor includes the OBC which updates the streaming order events in the message stream with financial market depth data in the same way, and for the reasons, as the Modified-OBS coprocessor described above because the modifications to Parsons's OBC for the OBS coprocessor do not

impact the manner in which Parsons's OBC enriches the streaming limit order events.

322. The TPS coprocessor includes the LVC which also updates the streaming order events in the message stream with updated price summary information, which includes (among numerous other derived price point values) updated price-aggregated trade information as discussed above. Parsons, [0060], [0062], [0095], [0127]-[0133].

323. Parsons does not disclose that every update the LVC computes is used to "enhance" (Parsons, [0095]) the message stream. However, Parsons discloses updating the message fields in the message stream that depend on record values the LVC updates, and gives "tick direction" as an example of a value that is maintained in the financial instrument records (Parsons, [0126]) and with which the message stream is updated. Parsons, [0124].

324. A POSA would have understood from this disclosure that updated values that the LVC computes that depend on the composite and/or regional record values that are accessed are used to modify the messages in the message stream to propagate the updates to downstream consumers. Thus, a POSA would have understood that the LVC updates limit order event messages in the message stream with at least the updated values it computes based on the limit order event message that depend on values stored in the accessed financial instrument records (e.g.,

derived values stored in the composite and regional price point records, like total trade volume at bid/ask, etc.), which are “*updated price point data*” computed by the LVC as discussed in VII.C.4.d. The volume values computed by the LVC such as total trade volume and volume-weighted average price (VWAP), and/or the price-aggregated trade values such total trade volume at bid/ask meet “*financial market depth data*” under any reasonable interpretation. In addition, these values also meet the ‘115 patent’s definition of “financial market data” at least because they are data derived from messages reporting a trade (i.e., “a completed sale of a financial instrument,” ‘115 patent, 1:66-67).

325. Each of the various price point, volume and other derived and price-aggregated trade information on trades on the exchanges disclosed as stored in the composite and regional records and updated in real-time are financial market depth data as each represents some aspect of state of the market for the corresponding financial instrument. Parsons, [0062] (“Each record represents the current state of that financial instrument in the market place”). Thus, the LVC updating the message fields of the corresponding limit order event message with the updated values it computed enriches “*the streaming limit order events with financial market depth data based on the computed updated [] price point data,*” as recited in limitation [43E].

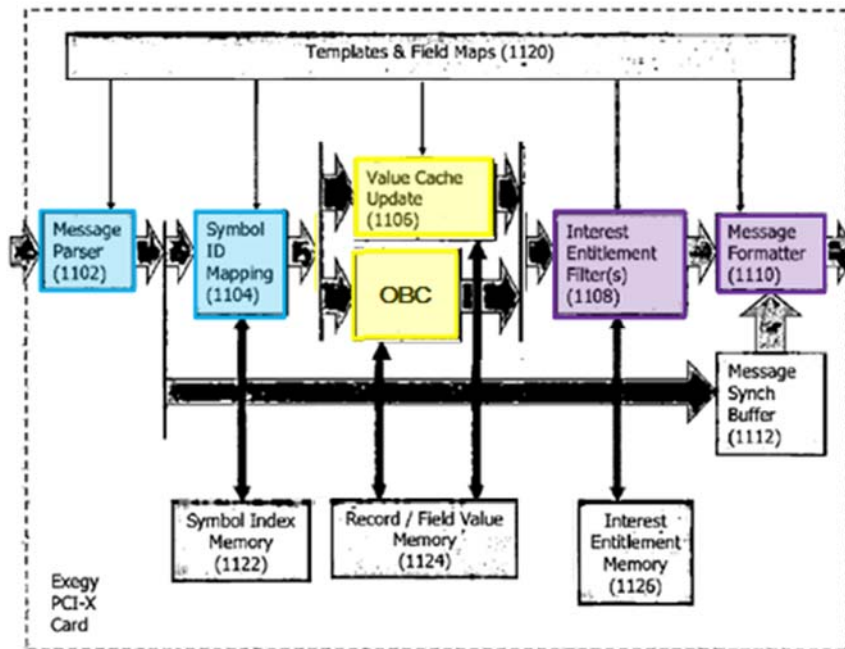
326. The Message Formatter of the TPS pipeline receives the updated message streams from the OBC and LVC and constructs an output message by applying the updates from the OBC and LVC to the original message stream by appending message fields from the OBC and LVC to the original message stream, replacing fields with those from the OBC and LVC, or a combination of both. Parsons, [0108], [0138].

327. The TPS coprocessor is therefore configured to “*enrich the streaming limit order events with financial market depth data based on the computed updated limit order data and price point data,*” as recited in [43E].

- g. [43F] “wherein the order engine and the price engine are configured to perform their respective operations in parallel with each other as the limit order event data streams through the coprocessor”**

328. The OBC and LVC (*i.e.*, the claimed “*order engine*” and “*price engine*,” respectively) perform their respective operations in parallel because they are arranged as parallel FAMs in the TPS pipeline.

329. As discussed in section VII.B.3, the TPS processor consolidates OBS and price summary functionality onto a single reconfigurable logic device that delivers both types of functionality, and arranges the OBC and the price LVC FAMS in parallel, as shown in Modified Figure 11 (below) to maintain the same parallel operation disclosed for market platform 600. *Id.*, [0056], Fig. 6.



**Figure 11
(Modified)**

330. The update operations performed by the OBC and LVC do not depend on each other and so are particularly amenable to this parallel arrangement. In fact, arranging the OBC and LVC in series would reduce throughput and increase latency, contrary to the teaching of Parsons. In the parallel arrangement use, the Message Parser parses limit order events reported in stream 106 from the exchanges common to both caching modules and delivers them to the OBC and LVC for parallel processing.

331. There are a number of ways in which memory for the caches can be allocated, as discussed below. Independent of the way in which memory could be allocated, the claimed “*respective operations*” of the order and price engines

recited in claim 43 are those in [43C] and [43D] that relate to accessing the engine's respective records and computing updates to them. In the TPS coprocessor the computing operations performed by the OBC's order engine and the TPC's price engine are done in parallel. Similarly, in one obvious implementation where the OBC uses a Memory Manager and record buffers, the OBC's order engine and the LVC's price engine access their respective records from different record buffers and do so in parallel. Thus, the claimed accessing and computing operations of the order and price engines are performed in parallel as required by [43F].

332. Claim 43 does not limit the order and price engines to accessing their respective records directly from cache, let alone in parallel directly from cache, and thus [43F] reads on the engines accessing their respective records from record buffers in parallel in the manner discussed above. With respect to how the records are accessed from cache in the TPS coprocessor, one way a POSA would have implemented the OBC and LVC caches by partitioning Record/Field Value Memory 1124 (e.g., on-board memory 404 in FIGS. 4(a), Parsons, [0060], [0071]) into separate memory spaces for the two caches, and by programming Symbol ID Mapping 1104 to map the financial instrument and exchange information extracted from the message stream to respective address pointers for the records maintained by each caching module. Parsons [0114]-[0124]. In this way, memory accesses to

the records maintained by the order and price engines are also performed in parallel. Thus, even if [43F] required that accesses to cache occur in parallel (it does not) the TPS coprocessor would meet [43F].

5. Claim 44

333. Claim 44 depends from claim 44 and further recites that the “*coprocessor comprises a member of the group consisting of **a reconfigurable logic device**, a chip-multi-processor (CMP), and a graphics processing unit (GPU), and wherein the order engine and the price engine are deployed on the member.*”

334. Like all of the devices disclosed by Parsons (*id.*, [0060], [0069]-[0070], [0163]-[0165], Figs. 2-6), the TPS is implemented on reconfigurable logic and a POSA would have been motivated to implement Parsons’s OBC and LVC in a single FAM pipeline loaded on a single reconfigurable logic device (e.g., a single FPGA).

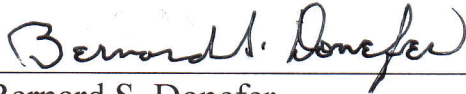
335. Thus, the TPS coprocessor meets claim 44 because it is implemented on an FPGA, which is a “*reconfigurable logic device*” that meets one of claim 44’s recited alternative members, and the TPS’s order and price engines are both “deployed” on the FPGA, and so meets “*the order engine and the price engine are deployed*” meets claim 44.

* * *

336. I understand and have been warned that willful false statements and the like are punishable by fine or imprisonment, or both (18 U.S.C. § 1001). I declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true, and further, that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under § 1001 of title 18 of the United States Code.

I declare under penalty of perjury that the foregoing is true and correct.

Dated: May 29, 2020



Bernard S. Donefer