Paper No. __

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. TBD
Patent No. 10,062,115

_____

PETITION FOR *INTER PARTES* REVIEW
UNDER 35 U.S.C. §§ 311-319 AND 37 C.F.R. § 42.1 *et seq.*

TABLE OF CONTENTS

I. INTRODUCTION 1

A. Background on Order Books 1

B. Purportedly Inventive Aspect of the Challenged Claims 3

C. Claims 43-44 Would Have Been Obvious Over Parsons 3

1. Ground 1A Summary 6

2. Ground 1B Summary 7

II. MANDATORY NOTICES 8

A. Real Party-In-Interest – § 42.8(b)(1) 8

B. Related Matters – § 42.8(b)(2) 9

1. United States Patent & Trademark Office 9

2. United States Patent Trial and Appeal Board 9

3. U.S. District Court for the Northern District of Illinois 9

C. Counsel and Service Information – §§ 42.8(b)(3) and (b)(4) 10

D. Certification of Grounds for Standing 10

E. Identification of Challenge and Relief Requested 10

III. BACKGROUND OF THE '115 11

A. '115 Overview 11

B. Level of Ordinary Skill in the Art 13

C. Discussion of the Challenged Claims 13

D. Claim Construction 15

IV. PARSONS RENDERS CLAIMS 43-44 OBVIOUS 16

A. Parsons (Ex. 1006) 16

1. Prior Art Status 16

2. Parsons's Disclosure 17

B. Ground 1A: Claims 43-44 are Rendered Obvious By Parsons's FPGA-Implemented Order Book Server (OBS) 20

1. The Obvious Implementation of Parsons's OBS 20

a. OBS Overview 20

b. The LVC Message/Record Updater 22

c. A POSA Would Have Been Motivated to Implement the OBC with the Parallel Processing Techniques Described For The LVC 24

d. Modified-OBS 27

2. An FPGA Implementing The Modified-OBS Renders Obvious Claim 43 27

a. [43PRE] "An apparatus for applying specific computer technology to reduce latency and increase throughput with respect to streaming data enrichment" 27

b. [43A] "a coprocessor that includes an order engine and a price engine" 28

i Coprocessor 28

ii Order and Price Engines 29

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" 31

d. *[43C.1]* "wherein the order engine is configured to (1) access a plurality of limit order records based on the streaming limit order event data" 33

e. [43C.2] "wherein the order engine is configured to… (2) compute updated limit order data based on the accessed limit order records and the streaming limit order event data" 36

f. [43D.1] "wherein the price engine is configured to (1) access a plurality of price point records based on the streaming limit order event data" 38

g. [43D.2] "wherein the price engine is configured to … (2) compute updated price point data based on the accessed price point records and the streaming limit order event data" 41

h. [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" 43

i. [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" 47
3. Claim 44 49
C. Ground 1B: Claims 43-44 Are Rendered Obvious By An Obvious Implementation Of Parsons Where The OBS and LVC Are Deployed On The Same FPGA 50
1. Background to Ground 1B 50
a. Parsons's Market Platform 50
b. A POSA Would Have Been Motivated to Implement Parsons's OBS and LVC on a Single "Ticker Plant" FPGA 53
c. Implementing the OBS on the Ticker Plant FAM Pipeline 56
2. Analysis of Challenged Claims 59
3. An FPGA Implementing The TPS Renders Obvious Claim 43 60
a. [43PRE] 60
b. [43A] 60
c. [43B] 61
d. [43C.1] 61
e. [43C.2] 63
f. [43D.1] 63
g. [43D.2] 65
h. [43E] 66
i. [43F] 69
4. Claim 44 72
D. Claims 43-44 Are Rendered Obvious Even If Recited Elements Are Considered to be § 112, ¶6 Elements 72
V. THE BOARD SHOULD INSTITUTE TRIAL 74
A. No Basis Exists To Deny Institution Under § 314(a) 74
B. No Basis Exists To Deny Institution Under § 325(d) 74

1. *Becton*, *Dickinson* Factor (c): the Petition's Art and Arguments Were Previously Not Evaluated for Claims 43-44.....75
2. *Becton*, *Dickinson* Factors (e)-(f): the Examiner Failed to Appreciate Parsons' Full Scope 77
3. Conclusion.....78

VI. CONCLUSION.....78

CLAIM LISTING 80

TABLE OF AUTHORITIES

CASES

Advanced Bionics v. Med-El Elektromedizinsche Gerate,
IPR2019-01469, Paper 6 (PTAB Feb. 13, 2020) 74, 75

Apple v. Fintiv,
IPR2020-00019, Paper 11 (PTAB Mar. 20, 2020).....74

Apple v. MPH Technologies Oy,
IPR2019-00819, Paper 10 (PTAB Sept. 27, 2019) 76, 78

Apple v. Omni Medsci,
IPR2020-00029, Paper 7 (PTAB April 22, 2020).....77

Asetek Danmark A/S v. Coolit Systems,
IPR2019-00705, Paper 19 (PTAB Sept. 6, 2019) 76

Becton Dickinson & Co. v. B. Braun Melsungen AG,
IPR2017-01586, Paper 8 (PTAB Dec. 15, 2017)..... 75, 77, 78

EmeraChem Holdings, LLC v. Volkswagen Group of Am., Inc.,
859 F.3d 1341 (Fed. Cir. 2017) 16

Exegy Incorporated et al. v. ACTIV Financial Systems, Inc.,
Case No. 1:19-cv-02858.....9

Fresenius USA v. Baxter Int'l,
582 F.3d 1288 (Fed. Cir. 2009) 49

In re Katz,
687 F.2d 450 (C.C.P.A. 1982).....16

KSR Int'l. Co. v. Teleflex Inc.,
550 U.S. 398 (2007) 25, 56

NHK Spring Co. v. Intri-Plex Techs.,
IPR2018-00752, Paper 8 (PTAB Sep. 12, 2018) 74

Puma North America v. Nike,
IPR2019-01058, Paper 10 (PTAB Oct. 31, 2019)..... 77, 78

Trans Ova Genetics v. XY,
IPR2018-00250, Paper 9 (PTAB June 27, 2018) 76, 78

Uniden Am. v. Escort,
IPR2019-00724, Paper 6 (PTAB Sept. 17, 2019) 74

Williamson v. Citrix Online, LLC,
792 F.3d 1139 (Fed. Cir. 2015) 72

STATUTES

35 U.S.C. § 102 16

35 U.S.C. § 102(a) 16

35 U.S.C. § 102(e) 16

35 U.S.C. § 103 10

35 U.S.C. § 112 72

35 U.S.C. § 122(b) 16

35 U.S.C. § 282(b) 15

35 U.S.C. § 314(a) 74

35 U.S.C. § 321(a) 10

35 U.S.C. § 325(d) 74

REGULATIONS

37 C.F.R. § 42.8(b)(1) 8

37 C.F.R. § 42.8(b)(2) 9

37 C.F.R. § 42.8(b)(3) 10

37 C.F.R. § 42.8(b)(4) 10

37 C.F.R. § 42.100(b) 16

37 C.F.R. § 42.101 10

37 C.F.R. § 42.104(a).....10

APPENDIX LISTING OF EXHIBITS

Exhibit	Description
1001	U.S. Patent No. 10,062,115 (“the ’115”)
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”)
1004	Declaration of Bernard Donefer (“Donefer”)
1005	Curriculum Vitae of Bernard Donefer (“Donefer CV”)
1006	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	*Newton’s Telecom Dictionary* (21st Ed. 2005) (“Newton’s Telecom ’05”)
1011	*Newton’s Telecom Dictionary* (21st Ed. 2009) (“Newton’s Telecom ’09”)
1012	Revised Case Management Schedule; *Exegy Inc. v. ACTIV Financial Systems, Inc.*, 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; *Exegy Inc. v. ACTIV Financial Systems, Inc.*, 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

I. INTRODUCTION

ACTIV Financial Systems, Inc. ("Petitioner") requests *inter partes* review of claims 43-44 of U.S. Patent No. 10,062,115 (Ex. 1001, "the '115").

The '115 is directed to high-speed processing of streaming data from financial exchanges (e.g., NYSE, NASDAQ). Financial exchanges provide a continuously updated stream of data reflecting recent orders (buy or sell) for financial instruments (e.g., stocks). One type of common order is a "limit order" which is "an offer to buy or sell a specified number of shares of a given financial instrument at a specified price" and thus limits the order to a specified price. Ex. 1001, 3:55-58.

A. Background on Order Books

To facilitate trading strategies it was conventional for traders to create (or be provided with) data structures that aggregated and organized the continuously updated streams of orders from financial exchanges to provide insight into market dynamics. *Id.*, 3:59-4:17. A set of such data structures was conventionally called an "order book."

Order books were conventionally updated from feeds provided by the financial exchanges which disseminate order "events" (e.g., addition, modification or deletion of limit orders). *Id.*, 3:48-61.

Speed is important in updating an order book as new order events are received because traders (e.g., those engaged in electronic trading) often seek to exploit market opportunities before they quickly dissipate. Declaration of Bernard Donefer ("Donefer," Ex. 1004) ¶ 46. The '115 is directed to a system for accelerating the updating of "order books." '115, Abstract, 4:63-5:14.

The '115's background describes two examples of data structures (shown in Figs. 2(a)-(b) reproduced below) conventionally included in order books. Fig. 2(a) shows a "full order depth" data structure that aggregates all outstanding orders for a financial instrument (e.g., stock). *Id.*, 4:3-6. Fig. 2(b) illustrates a "price aggregated depth" data structure that shows the number of outstanding orders for a financial instrument at each price. *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.

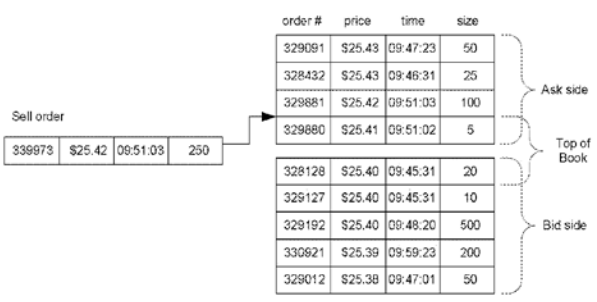

Figure 2(a)

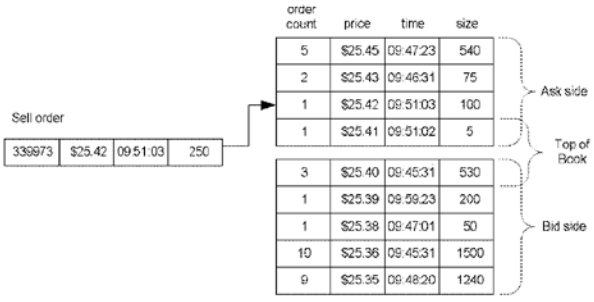

Figure 2(b)

B. Purportedly Inventive Aspect of the Challenged Claims

The ’115 concedes constructing and maintaining order books was known. 3:38-59; 4:63:5:1. The purported advance in claim 43 was to accelerate order book updating by: (1) implementing separate engines that operate ***in parallel*** to update the order book’s full order data structure (“order engine”) and price aggregation data structure (“price engine”), and (2) including the two engines on ***a coprocessor*** that offloads tasks from a system’s main processor.

Claim 44 requires that the coprocessor be one of three known devices, including a “reconfigurable logic device” which, per the specification, can be a field programmable gate array (FPGA). *Id.*, 3:5-7.

C. Claims 43-44 Would Have Been Obvious Over Parsons

Parsons (Ex. 1006), which shares two inventors with the ’115, is directed to a system that uses a hardware coprocessor (an FPGA) to “accelerate the speed of processing” of “financial market data.” Parsons, Abstract, [0039]. The coprocessor updates an order book that includes both a full order data structure and a price point aggregation data structure just like the ’115. *Id.*, [0071], [0096].

The hardware in Parsons and the ’115 is described almost identically. Donefer, ¶¶ 98-112. The ’115 explains that an FPGA is programmed using “firmware application modules (FAMs).” ’115, 10:13-11:42. Each FAM can be programmed to perform specific tasks that can be chained together in a “pipeline” to perform data processing operations. *Id.*; Donefer, ¶ 83. A FAM pipeline (850) on FPGA coprocessor (840) is illustrated in Figures 8(a) and 9(a) of the ’115 (annotated) below. *Id.*; Donefer, ¶ 83.

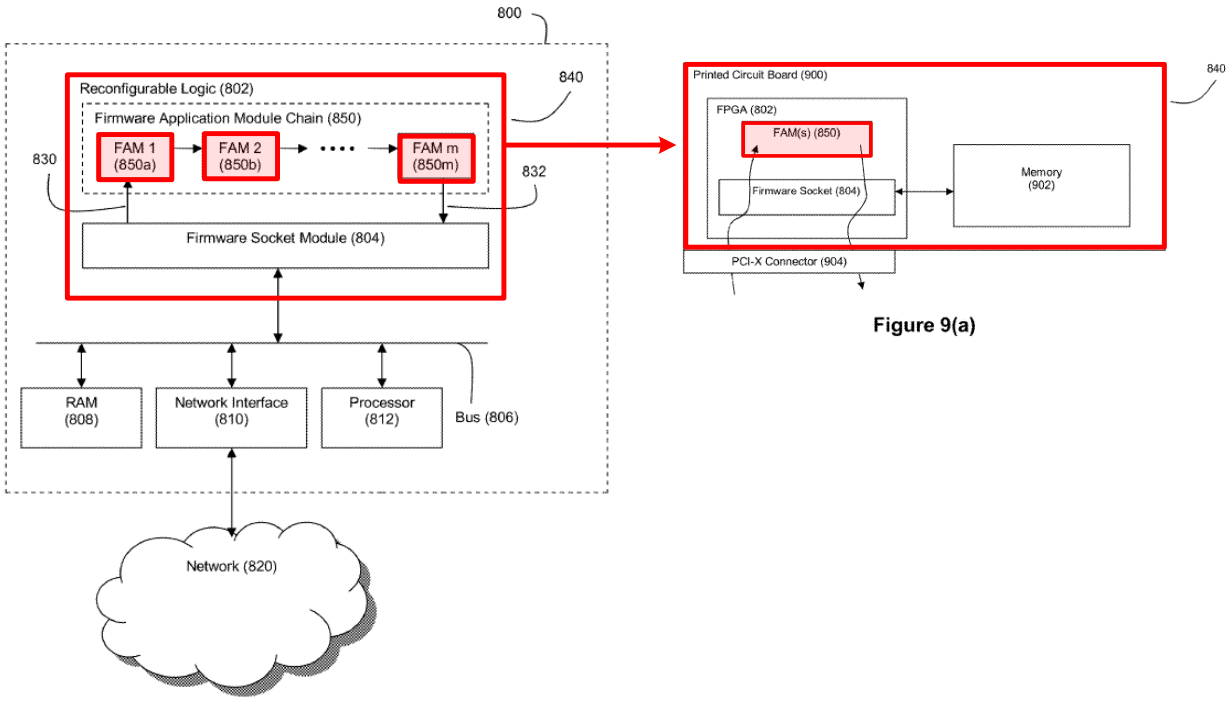

Figure 9(a)

Figure 8(a)

Figures 2 and 4(a) of Parsons, reproduced (and annotated) below, are almost identical to the ‘115’s Figures 8(a) and 9(a). Donefer, ¶ 103.

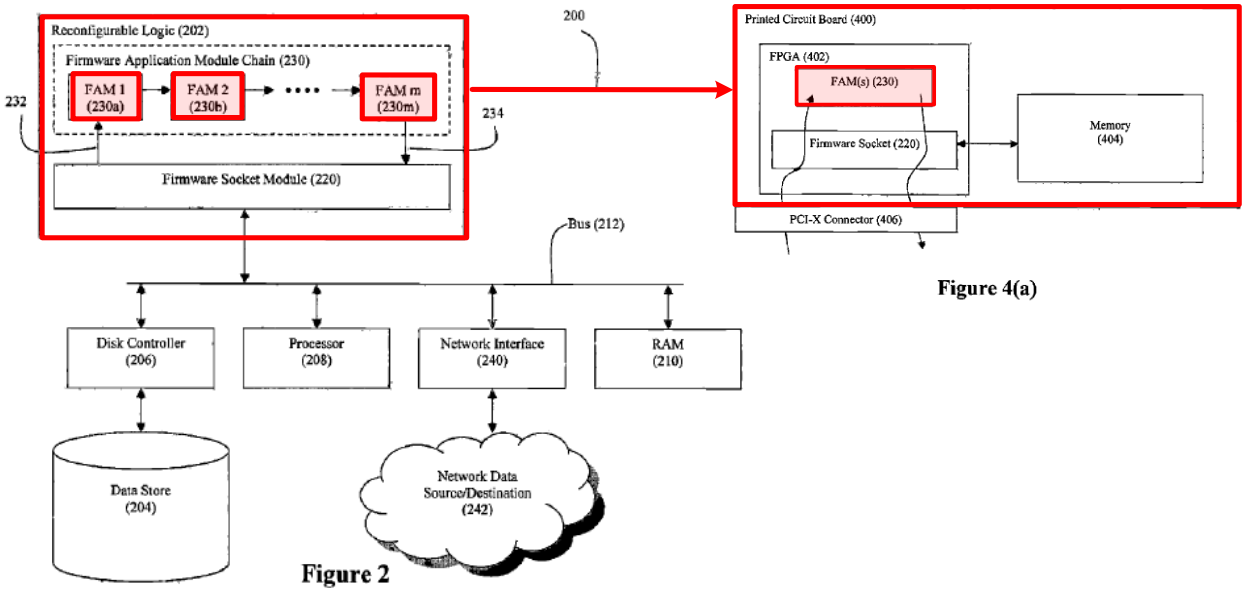

Figure 2

Figure 4(a)

In the ’115, the order engine that updates the full order data structure and the price engine that updates the price-aggregated data structure operate in parallel and are within the same coprocessor. ’115, 20:25-32. Parsons does not describe the two engines as being implemented both in parallel and on the same coprocessor. Thus, Parsons does not anticipate claims 43-44.

However, Parsons discloses an FPGA coprocessor with a FAM that updates both the order book’s full order data structure (an “order engine”) and its price-aggregated data structure (a “price engine”). Parsons, [0069]-[0071], [0096]. Although Parsons does not teach that those engines operate in parallel, a POSA would have had numerous reasons to configure Parsons’s FPGA in that manner. Doing so renders obvious claims 43-44 and that is the basis of Ground 1A summarized further below.

Parsons also discloses a different FPGA with a FAM that updates a different price point data structure and also meets the claimed price engine. *Id.*, [0060]-[0062], [0126]-[0127]. This different price engine operates in parallel with Parsons's above-described order engine (*id.*, [0056], FIG. 6), and a POSA would have had numerous reasons to implement these two engines on the same FPGA coprocessor. Doing so also renders obvious claims 43-44, and this is the basis of Ground 1B summarized further below.

1. Ground 1A Summary

Parsons discloses an FPGA configured to operate as an order book server (OBS) to maintain order book records for financial instruments based on a stream of limit orders received from exchanges. *Id.*, [0071], [0096]. Parsons's OBS also updates price point records. *Id*.

Parsons's OBS functionality is implemented by a FAM pipeline on an FPGA coprocessor. *Id.*, [0069]-[0071], [0077]. Parsons's OBS functionality that updates the limit order records (*id.*, [0071], [0096]) is an order engine as claimed, and the OBS functionality that updates the price point records (*id.*) is a price engine as claimed. Donefer, ¶¶ 144, 199-239.

Parsons does not describe the OBC's updating of order and price records as being performed in parallel. 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.

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. Parsons, Abstract; Donefer, ¶ 146. 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.

2. Ground 1B Summary

Parsons also discloses an LVC module that operates in parallel with the OBS. Parsons, [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. Donefer, ¶ 147.

In Parsons's implementation shown in FIG. 6, the LVC is implemented on device 604 and the OBS is implemented on a different device 606. Thus, the order

engine in the OBS and the price engine in the LVC are implemented on different FPGAs, and not "include[d]" on a single coprocessor as recited in claim 43.

For numerous reasons detailed below (§ IV.C.1.b), 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. Donefer, ¶¶ 276-287. This obvious modification to Parsons also renders claims 43-44 obvious.

II. MANDATORY NOTICES

A. Real Party-In-Interest – § 42.8(b)(1)

ACTIV Financial Systems, Inc. ("ACTIV" or "Petitioner") is the real party-in-interest.

IP Reservoir, LLC is a patent holding company that allegedly owns U.S. Patent No. 10,062,115. Exegy Incorporated ("Exegy") is a Delaware Corporation having its principal place of business at 349 Marshal Ave., Suite 100, St. Louis, MO 63119. Exegy is allegedly the exclusive licensee of the '115 within the field of processing financial market data. Ex. 1014. Exegy maintains the right to sue infringers of the '115 who make, use, offer for sale, sell, or import products and/or services that process real-time financial market data feeds that are created by or destined for financial market data exchanges. Exegy is a co-plaintiff in *Exegy*

Incorporated et al. v. ACTIV Financial Systems, Inc., Case No. 1:19-cv-02858, in which the '115 is asserted against Petitioner.

Petitioner identifies Exegy in an abundance of caution to assist the Board with respect to identification of any potential conflicts of interest.

B. Related Matters – § 42.8(b)(2)

A decision in this proceeding could affect or be affected by the following:

1. United States Patent & Trademark Office

The '115 is a divisional of U.S. Patent No. 8,768,805 ("the '805 patent"). The '805 patent and U.S. Patent No. 8,762,249 are both divisional applications of International App. No. PCT/US2009/067935.

Pending Application No. 16/111,530 claims the benefit of the filing date of the '115.

2. United States Patent Trial and Appeal Board

Petitioner is concurrently filing IPR petitions against U.S. Patent Nos. 7,921,046; 9,047,243 and 10,121,196. Petitioner requests that these petitions be reviewed by the same panel, as the patents are commonly assigned and contain overlapping subject matter.

3. U.S. District Court for the Northern District of Illinois

Exegy Inc. et al. v. ACTIV Financial Systems, Inc., Case No. 1:19-cv-02858.

C. Counsel and Service Information – §§ 42.8(b)(3) and (b)(4)

Lead Counsel	Richard F. Giunta, Reg. No. 36,149
Backup Counsel	Andrew J. Tibbetts, Reg. No. 65,139 Marc S. Johannes, Reg. No. 64,978 Michael N. Rader, Reg. No. 52,146
Service Information	E-mail: RGiunta-PTAB@WolfGreenfield.com ATibbetts-PTAB@WolfGreenfield.com MJohannes-PTAB@wolfgreenfield.com MRader-PTAB@wolfgreenfield.com Post and hand delivery: Wolf, Greenfield & Sacks, P.C. 600 Atlantic Avenue Boston, MA 02210-2206 Telephone: 617-646-8000 Facsimile: 617-646-8646

A power of attorney is submitted with the Petition. Counsel for Petitioner consents to service of all documents via electronic mail.

D. Certification of Grounds for Standing

Petitioner certifies that (1) the '115 is available for IPR; (2) Petitioner is not the owner of the '115; and (3) Petitioner is not barred or estopped from requesting IPR of claims 43-44. 35 U.S.C. § 321(a); 37 C.F.R. §§ 42.101, 42.104(a).

E. Identification of Challenge and Relief Requested

Petitioner requests cancellation of claims 43 and 44 on the following ground.

Ground Number and Reference(s)		Claims	Basis
1	Parsons (Ex. 1005)	43-44	§ 103

III. BACKGROUND OF THE ’115

A. ’115 Overview

The ’115 is directed to a coprocessor in a market data platform configured to perform order book construction and maintenance traditionally performed by downstream components. ’115, 3:38-5:14; Donefer, ¶¶ 78-92. This coprocessor can be implemented as an FPGA programmed via a pipeline of Firmware Application Modules (FAMs), as shown in annotated Figures 8(a) and 9(a) of the ’115 below. Donefer, ¶ 83.

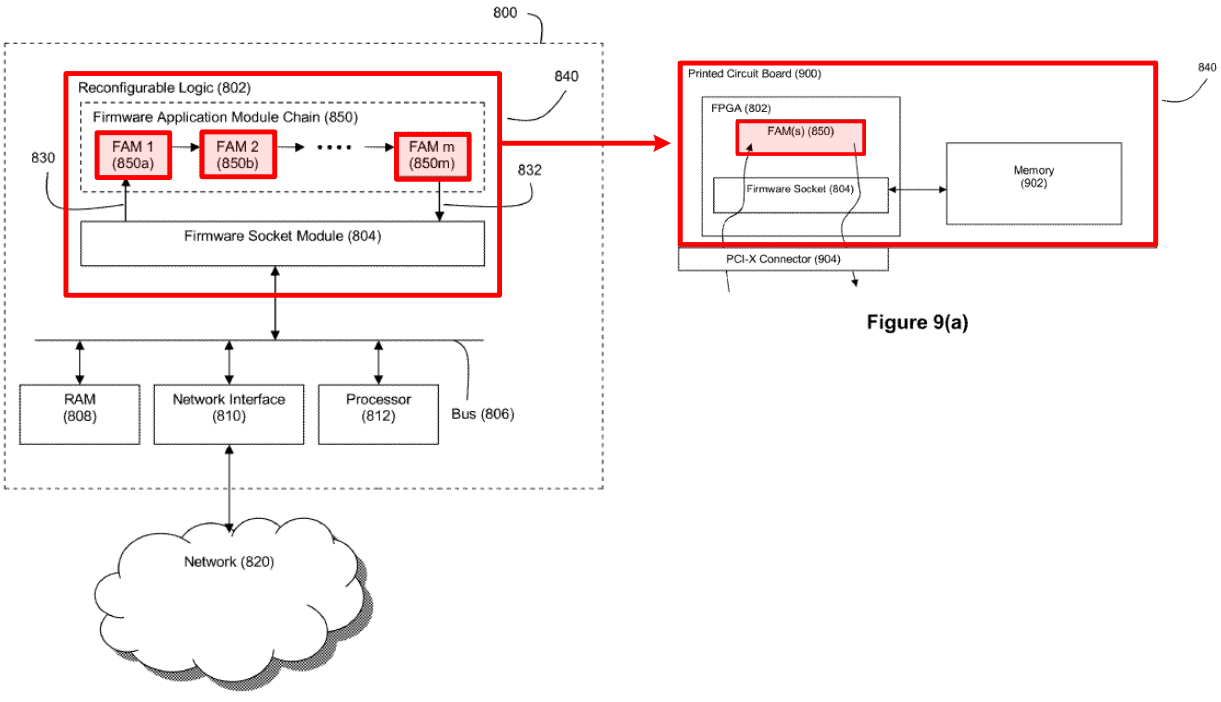

Figure 8(a)

Figure 9(a)

An example FAM pipeline configured to implement conventional order book functionality is illustrated in ‘115’s Figure 12(a):

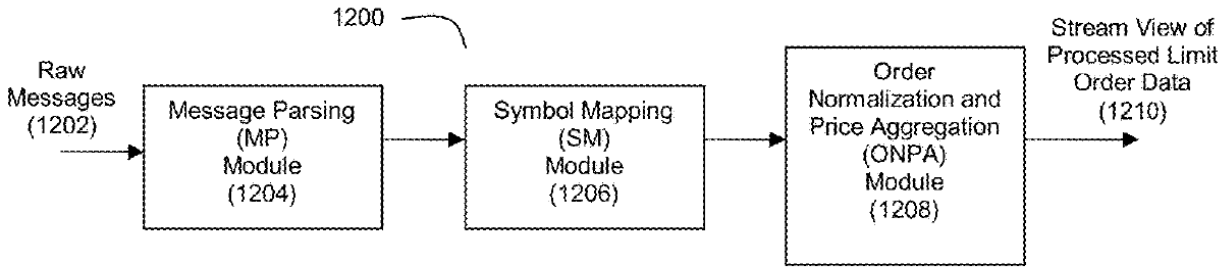

Figure 12(a)

Message Parsing Module (1204) parses a stream of financial market data into parsed messages having fields understandable to downstream components. '115, 12:25-38. Symbol Mapping Module (1206) maps information in the parsed messages to internal identifiers used to locate records stored in memory. *Id.*, 12:44-60, 14:65-15:7.

Order Normalization and Price Aggregation Module (ONPA) (1208) receives the parsed messages, updates corresponding limit order records and normalizes messages based on the received limit order events. *Id.*, 17:52-18:24. The stored limit order records are like the data structure in Fig. 2(a). Donefer, ¶ 86.

The ONPA additionally maintains price point records to support price aggregated views of the order book. '115, 18:6-20. In this data structure, a record is maintained for each unique price and includes fields for the total shares and orders at each price point. *Id.* These "price aggregated data structures" (20:26) are like the data structure in Fig. 2(b). Donefer, ¶ 87.

The ‘115 states that in a preferred embodiment “parallel engines update and maintain the order and price aggregation data structures in parallel.” ’115, 20:25-32. The ’115 refers to an engine that updates and maintains an order data structure as an “order engine,” and an engine that updates and maintains a price-aggregated data structure as a “price engine.” *Id.*; Donefer, ¶ 88.

B. Level of Ordinary Skill in the Art

A person of ordinary skill in the art (POSA) would have had (1) a bachelor of science 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. Additional education could have substituted for professional experience, and significant work experience could have substituted for formal education. Donefer, ¶¶ 93-95.

C. Discussion of the Challenged Claims

Independent claim 43 recites an apparatus that comprises a “coprocessor” “configured to receive streaming data representative of a plurality of limit order events.”

The coprocessor includes an ***order engine*** 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.

The coprocessor further includes a ***price engine*** 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.

The order and price engine limitations recite conventional operations performed on limit order information provided via market feeds. '115, 3:38-4:67; Donefer, ¶¶ 150-54. The '115 does not purport to have invented these operations, and acknowledges them to have conventionally been performed by "downstream components of a market platform" to construct order books. '115, 4:63-5:1.

The order engine maintains the "full order depth" data structure in FIG. 2(a), and the price engine maintains the "price aggregated depth" data structure in FIG. 2(b), both of which are acknowledged as conventional in the background. '115, 4:1-11; Donefer, ¶ 153.

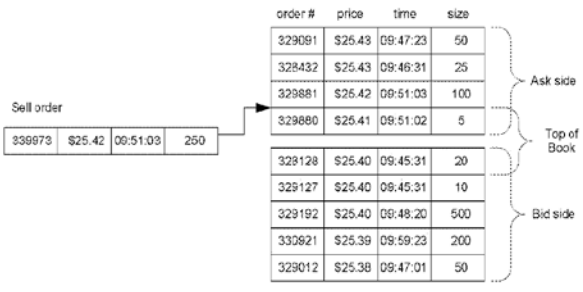

Figure 2(a)

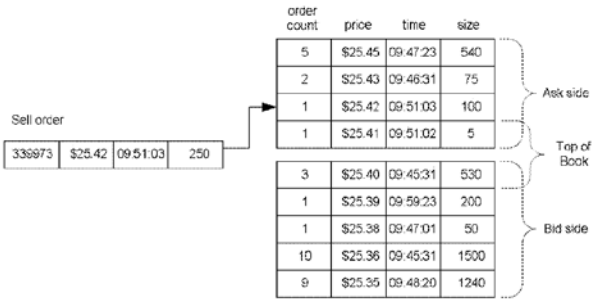

Figure 2(b)

Performing these operations on "a coprocessor that serves as an offload engine to accelerate the building of order books" ('115, 5:1-14) was also not new. Parsons disclosed ***precisely*** this solution using the same reconfigurable logic device (FPGA) to implement the coprocessor. Parsons, [0006]-[0015], [0071], [0096]; Donefer, ¶ 154.

Claim 43 further recites that the coprocessor "enriches the streaming limit order events with financial market depth data"—which simply means modifying or adding to limit order events with updated or additional information pertaining to the event, '115, 18:22-26; 18:61-63—the same function Parsons's FPGAs perform. Parsons, [0056], [0060]-[0062], [0069]-[0071], [0095]-[0096], [0100], [0124]-[0126], [0162].

Lastly, claim 43 recites the order and price engines as operating in parallel, which was the obvious way to implement Parsons's order and price engines to achieve the high processing throughput Parsons's desires. *Id.*, [0010]-[0015], [0096], [0132], [0162]; Donefer, ¶ 157.

D. Claim Construction

Claim terms are construed herein using the standard used in civil actions under 35 U.S.C. § 282(b). Terms defined in the specification are given their defined meanings, and other terms are construed in accordance with their ordinary

and customary meaning as understood by a POSA and the patent's prosecution history. 37 C.F.R. § 42.100(b).

IV. PARSONS RENDERS CLAIMS 43-44 OBVIOUS

A. Parsons (Ex. 1006)

1. Prior Art Status

Pre-AIA § 102 applies because the '115 claims priority to applications filed before March 16, 2013. Pub. L. No. 112-29, § 3(c), 125 Stat. 284, 293 (2011). Parsons is prior art under pre-AIA § 102(a) because it published October 2, 2008, prior to the '115's earliest possible priority date of December 15, 2008, and it describes the work of another. *In re Katz*, 687 F.2d 450, 454 (C.C.P.A. 1982) (explaining that a printed publication can only stand as prior art under § 102(a) if it describes "the work of another."). Parsons is also prior art under pre-AIA § 102(e) because it is an application for patent by another that was both published under § 122(b) and granted as a patent, and that was filed in the U.S. (on June 19, 2007) before the earliest possible priority date of December 15, 2008.

Parsons is by "another" under §§ 102(a) and 102(e) because it lists different inventors than the '115. *In re Katz*, 687 F.2d at 454; *EmeraChem Holdings, LLC v. Volkswagen Group of Am., Inc.*, 859 F.3d 1341 (Fed. Cir. 2017). Parsons lists five co-inventors. Only two of those co-inventors (Parsons and Taylor) are listed as inventors on the '115.

2. Parsons's Disclosure

Parsons is directed to consolidating financial trading operations on "reconfigurable logic, such as Field Programmable Gate Arrays (FPGAs)" to process financial market information at "hardware speeds" and reduce latencies. Parsons, [0010]-[0011]; Donefer, ¶ 98. Conventional market data platforms distributed financial market data processing over a network of individual computers—functional units 102 boxed in FIG. 1 below. Parsons, [0006].

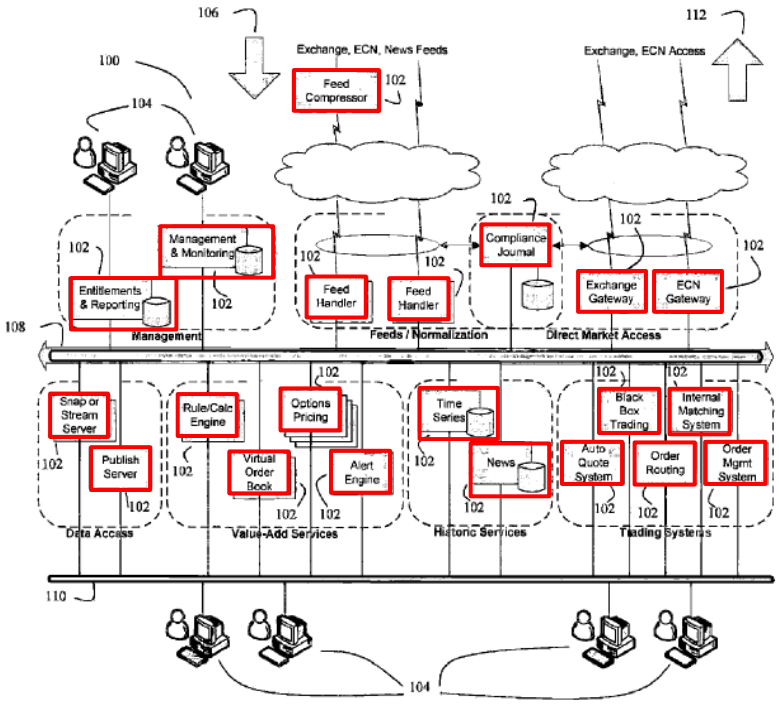

Figure 1
PRIOR ART

This platform introduced software latencies and transmission delays in communications between the multiple computers. *Id.*, [0008]. Parsons discloses a hardware-accelerated platform that "offloads" the same functionality onto a fewer number of reconfigurable logic devices (RLDs). *Id.*, [0056].

Like the '115, Parsons configures individual FAMs to perform data processing tasks that can be loaded onto reconfigurable logic and chained together in a pipeline, as shown in annotated Figure 2 below. *Id.*, [0045]; Donefer, ¶¶ 101-104.

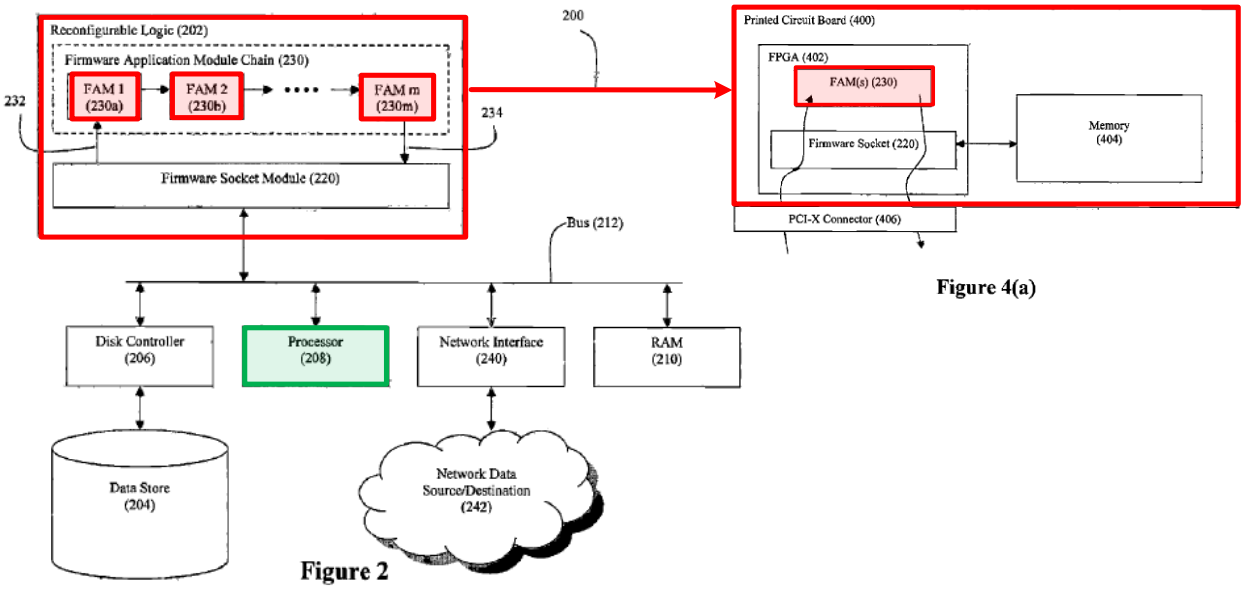

Figure 2

Figure 4(a)

This reconfigurable logic can be implemented on FPGAs connected to "a system's main processor 208" (green above) to offload computations from the main processor. Parsons, [0039], [0052], [0056]; Donefer, ¶ 102. Fig. 6 illustrates Parsons's market platform 600 that implements financial data processing on RLDs, including devices 604 and 606 highlighted and annotated below.

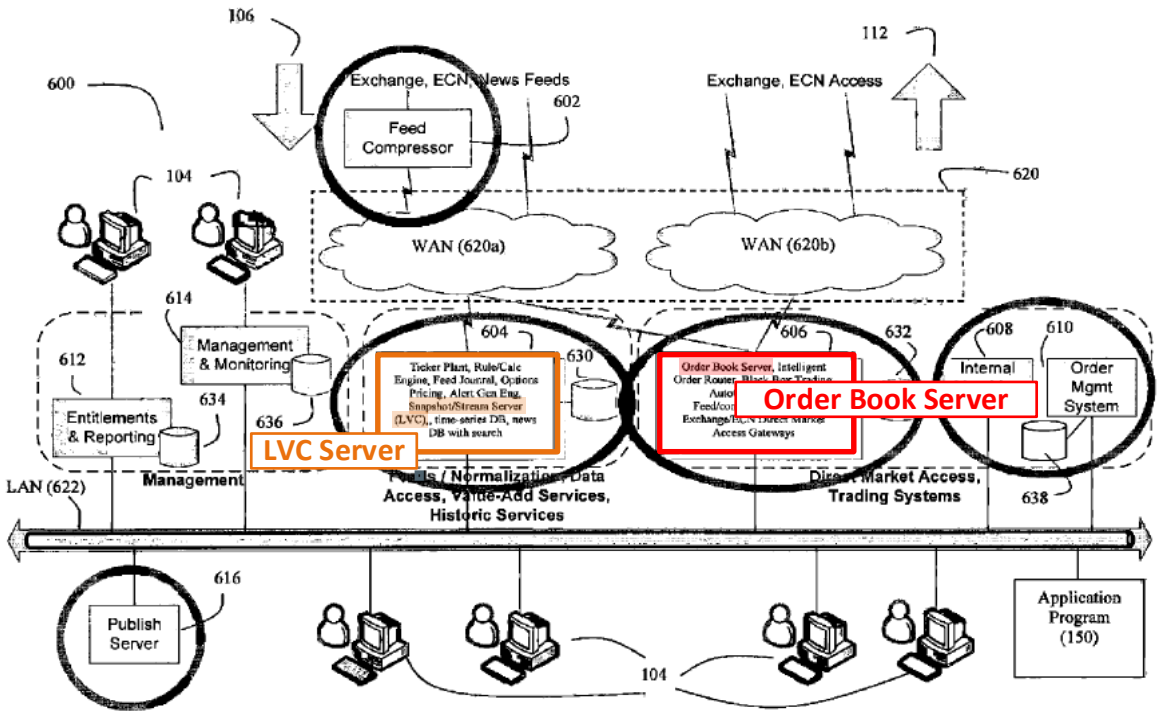

Figure 6

RLDs 604 and 606 provide 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 exchanges and provide their respective outputs to downstream traders at workstations 104. Parsons, [0056]-[0062], [0071]; Donefer, ¶¶ 105-06.

The OBS and LVC servers employ respective caching modules each configured to maintain a cache of financial instrument records as updates to those financial instruments are reported by exchanges. Each caching module is programmed to perform specific updates needed to keep data structures stored by that caching module up-to-date. Parsons, [0060], [0062], [0071], [0095]-[0096], [0125]-[0132]; Donefer, ¶¶ 107-08.

To maintain up-to-date order books, the OBS employs an "order book cache update, retrieve and normalize (OBC)" FAM to update limit order records and price aggregation records for each financial instrument record stored in its cache. Parsons, [0071], [0096]. The OBC's reconfigurable logic configured to perform updates to the order and price-aggregated data structures meets the claimed "order engine" and "price engine," respectively. Donefer, ¶¶ 112-116.

The LVC server uses an LVC module (also referred to as a "price summary LVC," Parsons, [0095]), one example of which is configured to maintain regional and composite 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 reconfigurable logic that maintains these regional and composite records also meets the claimed "price engine." Donefer, ¶ 110.

B. Ground 1A: Claims 43-44 are Rendered Obvious By Parsons's FPGA-Implemented Order Book Server (OBS)

1. The Obvious Implementation of Parsons's OBS

a. OBS Overview

Parsons's OBS is implemented on device 606, which is an FPGA (402) mounted on printed circuit board (PCB) 400 and having FAM pipeline (230) loaded thereon as shown in annotated Figure 4(a) below. Parsons, [0045], [0052]-[0053], [0056], [0069]-[0071]; Donefer, ¶¶ 117, 127-134.

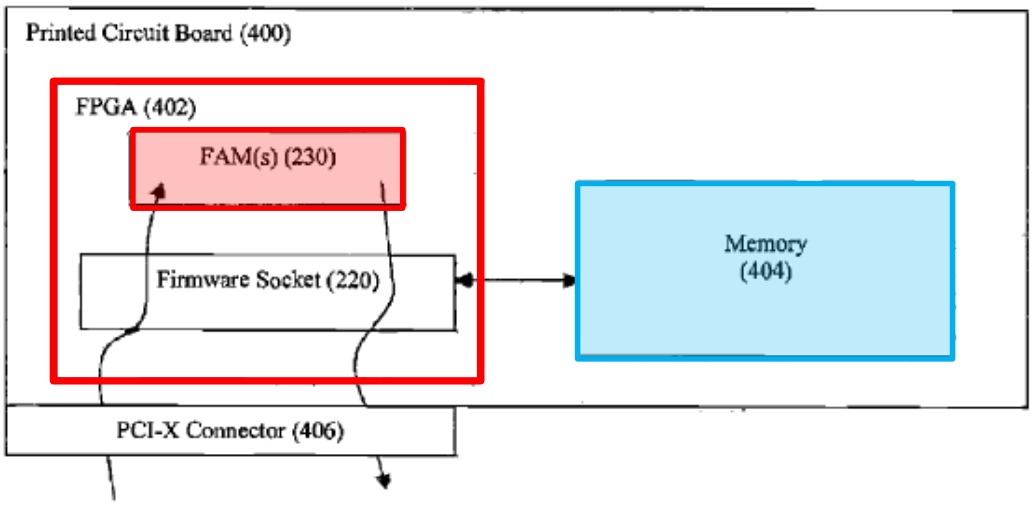

Figure 4(a)

The OBS pipeline receives data stream 106 to maintain order books for financial instruments. Parsons, [0071], [0077], [0096]. Order books are stored as "financial instrument records" in memory 404 (highlighted blue) on PCB 400. To access and update order books in real-time, the OBS pipeline implements the OBC as a caching module to update the financial instrument records. *Id.*, [0006], [0071], [0096]; Donefer, ¶ 118.

The OBC's cache of financial instrument records includes 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." Parsons, [0071], [0096]; Donefer, ¶¶ 119-121. The order and price-aggregated data structures for a financial instrument are

referred to as "sub-records" because the OBC maintains them as parts of a larger record for that financial instrument. *Id.*

The OBC updates its sub-records 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." Parsons, [0096]; *see also id*. [0006], [0071]; Donefer, ¶ 126.

b. The LVC Message/Record Updater

Parsons's LVC includes a Memory Manager configured to retrieve records from the LVC's cache 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. It then retrieves and loads records for the next message in the stream so that multiple messages can be processed in parallel to achieve higher message throughput. Parsons, [0125]-[0134], Fig. 15(a); Donefer, ¶ 135.

In Fig. 11, the LVC Memory Manager retrieves composite and regional records corresponding to the financial instrument identified in a message and loads those records into corresponding record buffers, as shown in annotated Fig. 15(a) below. Parsons, [0126]-[0127], [0132], Fig. 15(a); Donefer, ¶¶ 136-39.

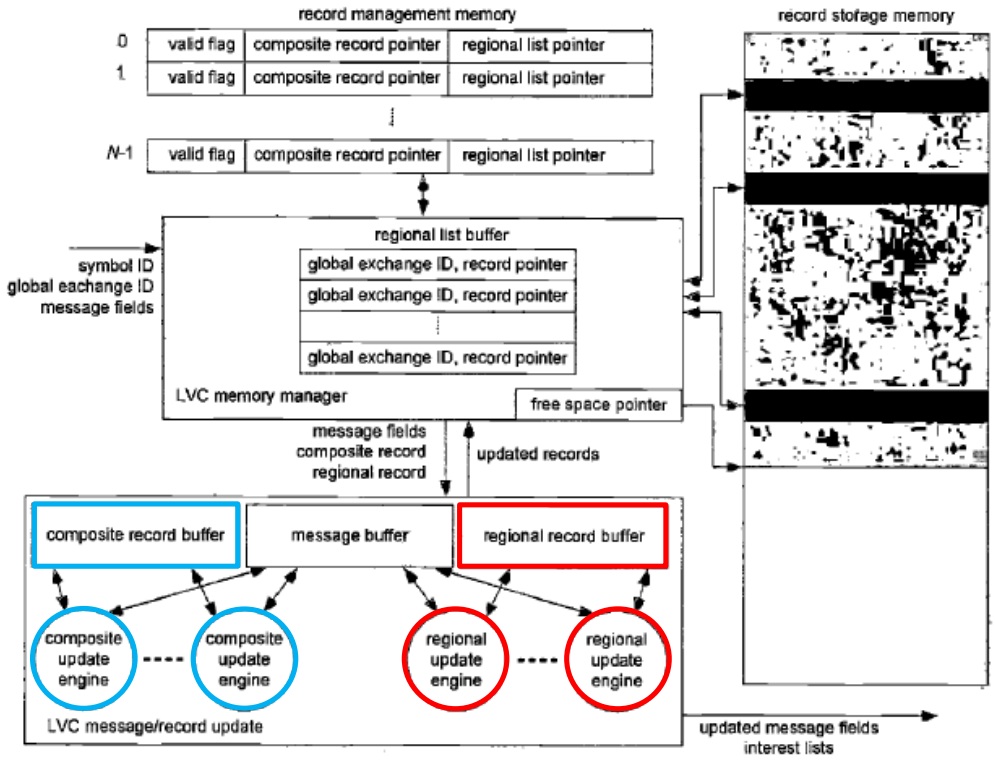

Figure 15(a)

When different record types are maintained and updated, the processing may be distributed by assigning separate update engines the task of updating a particular record, so that multiple records are updated by different engines operating in parallel. Donefer, ¶¶ 136-39.

As illustrated in annotated Fig. 15(a) above, the LVC includes a message/record updater that 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. Parsons, [0125], [0132]; Donefer, ¶¶ 140-41.

Thus, the LVC's message/record updater assigns the task of updating each retrieved data record in the record buffers to a different update engine (or group of update engines) which distributes the processing load required to update the records across different update engines and achieves "high throughput." Donefer, ¶¶ 140-41.

c. A POSA Would Have Been Motivated to Implement the OBC with the Parallel Processing Techniques Described For The LVC

The OBC performs cache accesses based on a record key computed from the financial instrument identified in a message in the message stream (Parsons, [0093]), like Parsons's other caching modules (including the LVC). *Id.*, [0093]; Donefer, ¶ 158. Parsons provides no further detail on how the OBC functionality that updates its data structures interfaces with the OBC's cache. *Id.*

A POSA would have had reasons to implement the OBC's cache interface using a Memory Manager, record buffers, and a message/record updater as Parsons teaches for the LVC. Parsons, [0125]-[0134]; Donefer, ¶ 158. The OBC and LVC are both caching modules that maintain and update financial instrument records stored in a cache. Parsons, [0095]-[0096]; Donefer, ¶ 158. A POSA would have

understood that the OBC would benefit from using the same record updating techniques Parsons describes for the LVC. Donefer, ¶¶ 158-170.

A POSA would have been motivated to implement the OBC to use the parallel processing techniques Parsons describes for the LVC so that multiple records could be updated in parallel to achieve "high throughput." Parsons, [0132]; Donefer, ¶ 159. A POSA would have been so motivated given that the OBC has multiple record types (i.e., the order sub-records and the price-aggregated sub-records) to update in real-time based upon the same received message (e.g., a limit order message), and given that the OBC's operations are "latency sensitive." Parsons, [0071], [0096]; *see also, id.*, [0010]-[0015], [0071], [0096], [0162]; Donefer, ¶¶ 159, 160-66.

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. Donefer, ¶¶ 167; *KSR Int'l. Co. v. Teleflex Inc.*, 550 U.S. 398, 418 (2007).

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 update operations by record type as Parsons describes for the LVC.

Donefer, ¶¶ 165-168. When implemented in this manner, the OBC employs an order engine (that updates the order records) and a price engine (that updates the price-aggregated records) operating in parallel as shown below to perform the update operations Parsons describes of the OBC. *Id.*, ¶ 168.

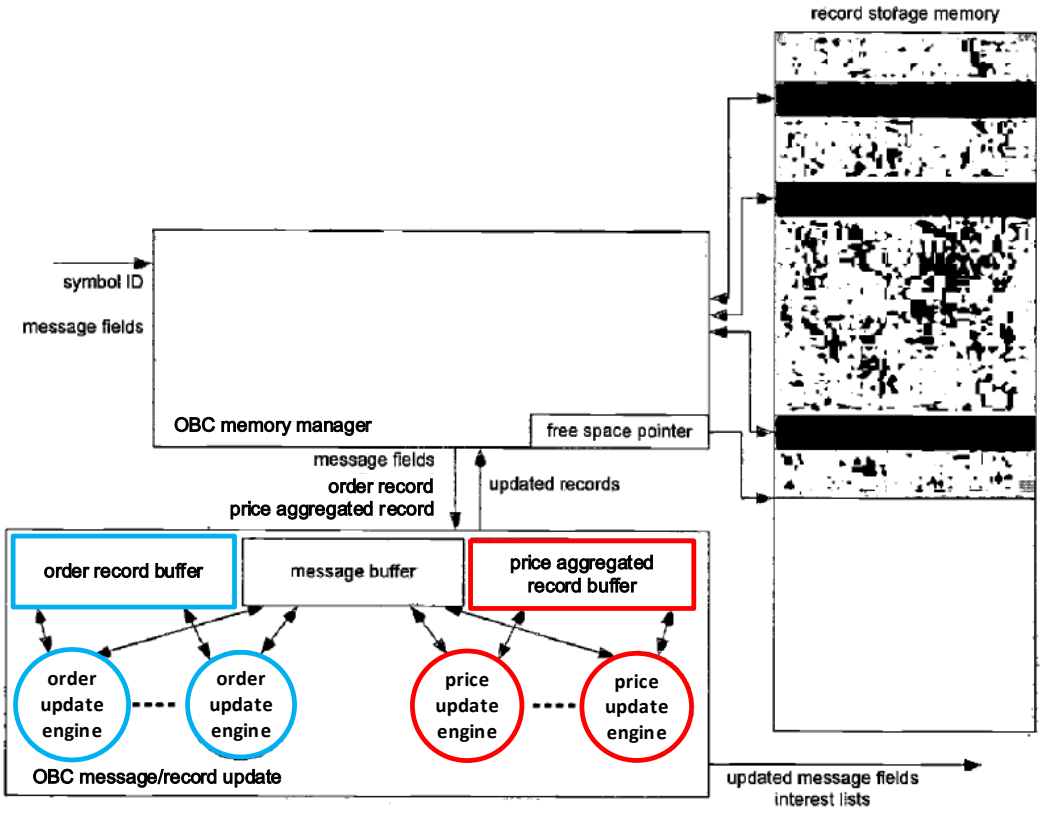

Figure 15(a)
(Modified)

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], Donefer, ¶ 168. It also

would have been obvious to use a single update engine to implement the order and/or price engine in the OBC. *Id*.

The above-discussed implementation of the OBC, which distributes order record updates across one or more "order" update engines, and price-aggregated record updates across one or more "price" update engines, so that order records and price-aggregated records are accessed from their corresponding record buffers and updated in parallel, is referred to herein as the "Modified-OBC." Donefer, ¶ 169. Parsons's Order Book Server (OBS) implemented using the Modified-OBC is referred to herein as the "Modified-OBS." Donefer, ¶¶ 169-170

d. Modified-OBS

An FPGA implementing the Modified-OBS meets every limitation of claims 43-44 as detailed below and renders those claims obvious. *Id*., ¶ 170.

2. An FPGA Implementing The Modified-OBS Renders Obvious Claim 43

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

If the preamble of claim 43 is deemed limiting it is met by an FPGA implementing the Modified-OBS, which is an apparatus applying "*specific computer technology*" at least per claim 44. Parsons's FPGAs process "vast amounts of streaming financial information" at "hardware speeds" to enrich the

streaming data at increased throughput with "less latency." Parsons, [0008], [0010]-[0011], [0095] (message stream is processed to "enhance" the messages), [0096], [0162]; Donefer, ¶¶ 171-174; *see also* below re limitation [43E] for enriching.

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

i Coprocessor

An FPGA implementing Parsons's Modified-OBS (hereafter "Modified-OBS coprocessor") is a coprocessor as claimed. Donefer, ¶¶ 176-179. The Modified-OBS may be implemented on device 606 in Fig. 6 just like Parsons describes of the OBS. *Id.*, ¶ 176. Device 606 may be deployed in an appliance such as system 200 in Fig. 2. Parsons, [0069]. The reconfigurable logic 202 of system 200 may be implemented on PCB board 400 as shown in Figs. 4(a)-(b). *Id.*, [0069]-[0071]; § IV.B.1. Figures 2 and 4(a) are annotated together below.

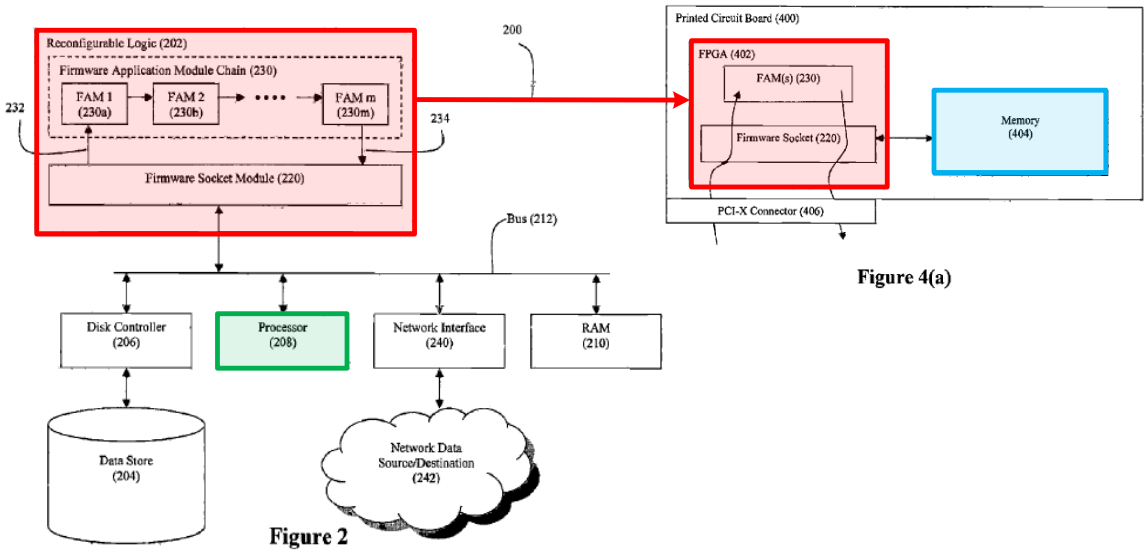

Figure 4(a)

Figure 2

Device 202 receives data from "disk controller 206 and data store 204" and is connected to "the computer system's main processor 208 … [and] RAM 210" (Parsons, [0039]), and is therefore a "coprocessor" according to the '115's definition because it is "a computational engine designed to operate in conjunction with other components in a computational system having a main processor." '115, 2:58-67; Donefer, ¶¶ 177-178.

Parsons's device 606 offloads tasks from the main processor as the '115 indicates coprocessors "typically" do. *Id.*; Parsons, [0056] (market platform 600 "***offloads*** much of the data processing performed by the GPPs of the functional units 102" of Fig. 1). Parsons does not use the term "coprocessor" to describe its FPGA devices, but the '673 provisional incorporated by reference into the '115 (1:12-16) explicitly characterizes Parsons's FPGA device as a coprocessor. Ex. 1003, Page 37.

Thus, an FPGA implementing the Modified-OBS meets the claimed "coprocessor." Donefer, ¶ 179.

ii Order and Price Engines

The '115 uses "engine" to describe a range of computational elements of the coprocessor and of other elements, e.g., "exchange matching engines," "basket calculation engines," "compression engines," "sorting engines," "hash engine," etc. '115, 2:58-594:5-6, 9:45, 13:13-14, 18:27, 35:65. Thus, "engine" in the '115 is a

set of computational elements that perform a particular function. Donefer, ¶ 180. 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. *Id*. The Modified-OBC meets the claimed order and price engines. *Id.*, ¶ 181.

The financial instrument records maintained by Parsons's Modified-OBC store 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]; Donefer, ¶ 182. This array of sub-records is updated in real-time based on order information in the message stream to keep the order books up-to-date. *Id.*

The computational elements in Parsons's Modified-OBC that update the order sub-records meet the claimed "order engine" because they meet each of the limitations in [43C], as discussed in §§ IV.B.2.d-e below, and are implemented (by an FAM on an FPGA) in the same manner as the "order engine" described in the '115. Donefer, ¶¶ 183-185.

The computational elements in Parsons's Modified-OBC that update the price-aggregated sub-records similarly meet the claimed "price engine" because they meet the limitations in [43D], as discussed in detail in §§ IV.B.2.f-g below,

and are implemented (by an FPGA FAM) in the same manner as the "price engine" described in the '115. Donefer, ¶¶ 186-187.

This mapping is consistent with the '115, which maps the "order engine" and "price engine" to the computational elements that update and maintain the order and price aggregation data structures, respectively. '115, 20:25-30; Donefer, ¶ 189.

An FPGA with the reconfigurable logic (FAM pipeline) of Parsons's Modified-OBS includes the FAM that implements the Modified-OBC (§ IV.B.1.c above), and thus this FPGA (the claimed coprocessor) "includes" the order and price engines (implemented by the Modified-OBC) as claimed. Donefer, ¶ 190.

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"

The Modified-OBS coprocessor receives "financial data stream 106" (also called "feed stream 106") from exchanges reporting limit order events (e.g., "bid, offer and trade information," [0006]) for a plurality of financial instruments. Parsons, [0006] ("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"); [0071], FIG. 1; Donefer, ¶¶ 191-192.

Parsons does not use the term "limit order" but a POSA would have understood Parsons's bids and offers to comprise limit orders for a host of reasons. Donefer, ¶¶ 192-195. First, Parsons's description of financial data stream 106 (Parsons, [0006]) is used verbatim in the '115 to define "financial market data" ('115, 1:63-2:2), which the '115 explains includes limit order events. *Id*., 3:38-4:67. Donefer, ¶¶ 193, 196-198.

Second, Parsons's OBS uses the "order information for each financial instrument [] received … in stream 106" to maintain "a sorted list of ***bids*** and ***offers*** associated with all ***outstanding orders***" for that financial instrument. Parsons, [0071]; Donefer, ¶ 195. These outstanding orders are ***limit*** orders because a "bid" specifies the maximum price a buyer is willing to pay for a specified number of shares, and an "offer" (also called an "ask") specifies the minimum price at which a seller is willing to sell a specified number of shares. '115, 3:55-59 ("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."); Donefer, ¶ 195.

Third, Parsons maintains order books with orders sorted by price, and price-aggregated records of orders for a plurality of financial instruments at various prices; those price-limited orders are limit orders as claimed. *Id.*; Parsons, [0096].

Thus, the Modified-OBS coprocessor meets [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.*" Donefer, ¶ 198.

d. *[43C.1]* "wherein the order engine is configured to (1) access a plurality of limit order records based on the streaming limit order event data"

Limitation [43C.1] recites "*the streaming limit order event data,*" which refers to the "*streaming data representative of a plurality of limit order events*" recited in [43B]. *Id.*, ¶ 200. Parsons's Modified-OBS pipeline processes this order information using a parsing module that parses the stream into its constituent messages and provides them as a "message stream" to the OBC FAM to update the order books. Parsons, [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."); Donefer, ¶¶ 128, 131, 200.

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. Parsons, [0071], [0096]; Donefer, ¶ 200.

As discussed above, the 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. Donefer, ¶¶ 201-202; § IV.B.2.c; Parsons, [0096] (outstanding orders are sorted by ***price***).

Each financial instrument record includes an array of sub-records that "define" individual order entries for ***each*** outstanding order for that financial instrument. Parsons, [0096]; Donefer, ¶¶ 182, 203, 228. This data structure maintained by the Modified-OBC serves the same purpose as the "full order depth" data structure of the conventional order book in Figure 2(a) of the '115 (below). Donefer, ¶ 204.

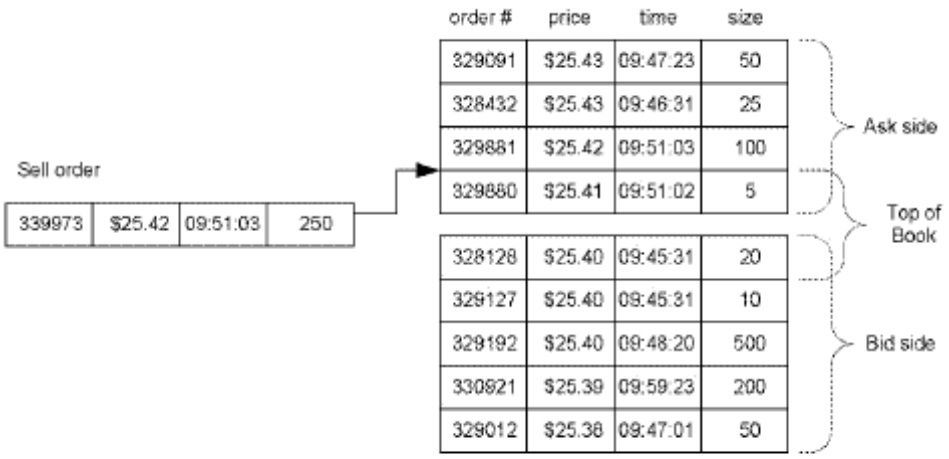

Figure 2(a)

Claim 43 does not require that the "*plurality of limit order records*" have any particular structure or particular fields. '115, 17:25-27 (a limit order record can be "configured to have ***more or fewer and/or different data fields***" than the example limit order record in FIG. 20); Donefer, ¶ 205.

Thus, the Modified-OBC's sub-records of a financial instrument record 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]; Donefer, ¶ 206.

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 memory. Donefer, ¶ 207. The Modified-OBC uses a direct record key number mapped from the symbol for each financial instrument to directly address the corresponding financial instrument record in cache. Parsons, [0093]-[0094]. The Modified-OBC's Memory Manager retrieves order sub-records from the addressed financial instrument record and loads them into corresponding record buffers. Parsons, [0125]-[0132]; Donefer, ¶ 207; § IV.B.1.c.

The Modified-OBC's update engine(s) assigned to perform updates to the order sub-records access those sub-records from the record buffer to perform updates according to the information in the message stream. Parsons, [0125]-[0131]-[0132]; Donefer, ¶ 208; § IV.B.1.c.

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 are those whose fields are updated based on the order information

(i.e., "*limit order event data*") extracted from the message stream. Parsons, [0071], [0096]; Donefer, ¶¶ 209-211.

e. [43C.2] "wherein the order engine is configured to… (2) compute updated limit order data based on the accessed limit order records and the streaming limit order event data"

The Modified-OBS maintains order books based on messages reported from exchanges, and does so by updating, creating and removing sub-records to keep the order books up-to-date in real-time. Parsons, [0006], [0071], [0096]; Donefer, ¶¶ 212-13.

The operations to keep order books up-to-date based on update messages from exchanges were well-known. Donefer, ¶ 210; Korhammer, 9:27-49 (describing updates in response to new order, change order and delete order events.). The '115 acknowledges this. '115, 3:50-4:67. Specifically, (1) new limit order records are added when new limit order events are reported, (2) records are removed when a delete order event reflects an order has been fully satisfied, and (3) existing limit order records are modified to reflect trades on a limit order for less than the full order size. Donefer, ¶ 210; Korhammer, 9:27-49.

Parsons's Modified-OBC is configured to perform all three of these types of update operations, and performing any one of them individually, and all of them collectively, 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"); Donefer, ¶¶ 214-223.

The '115 does not use the term "*updated limit order data*" but "*limit order data*" is used to describe data in the input stream. '115, 4:41, 6:51-7:26, 9:62, 12:11-34; Donefer, ¶ 219. The operations the '115 describes (performed by its ONPA module) on an accessed limit order record include updating the record and providing the update information to the message stream for access by downstream subscribers. '115, 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); Donefer, ¶ 219.

Parsons performs the same operations; the OBC updates fields of the order sub-records (Parsons, [0096]) in the cache as discussed above, and also enriches the message stream with the update information (based on the accessed records and the new order message) as discussed further below in connection with limitation [43E]. *Id.*, [0100]; Donefer, ¶¶ 212-18. Parsons's Modified-OBC performing the same types of operations on an accessed limit order record as disclosed in the '115 reinforces that the Modified-OBC meets limitation [43C.2]. *Id.*

As one additional example of how [43C.2] is met, a POSA would have understood the Modified-OBC computes a new size 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. Parsons, [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]; Donefer, ¶¶ 216, 222. 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. *Id*. Computing a new order book record for a partially fulfilled order meets limitation [43C.2].

f. [43D.1] "wherein the price engine is configured to (1) access a plurality of price point records based on the streaming limit order event data"

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 describes its price aggregation functionality in almost identical terms. '115, 18:6-16; Donefer, ¶¶ 224-27.

The array of sub-records for each financial instrument record maintained by Parsons's Modified-OBC store price-aggregated entries for that financial instrument; each financial instrument record "consists of an array of sub-records that define individual ***price*** or ***order*** entries for that financial instrument." Parsons,

[0096]; Donefer, ¶ 225. That is, each sub-record in the array stores either an order entry or a price entry. Donefer, ¶ 225. The price sub-records support price-aggregated views and correspond to the above-discussed aggregated price point "entries that are indicative of the total number of orders available at each price point." Parsons, [0071], [0096]; Donefer, ¶ 225.

Thus, the price-aggregated entries maintained by the Modified-OBC store the ***same*** information as the '115's price point records (also called "price aggregation data structures," '115, 20:25-26), i.e., aggregated orders at each price point. '115, 18:6-16; Donefer, ¶ 226. The price-aggregated data structure maintained by the Modified-OBC is similar to the "price-aggregated depth" data structure of the conventional order book in Figure 2(b) of the '115 (below). Donefer, ¶ 227.

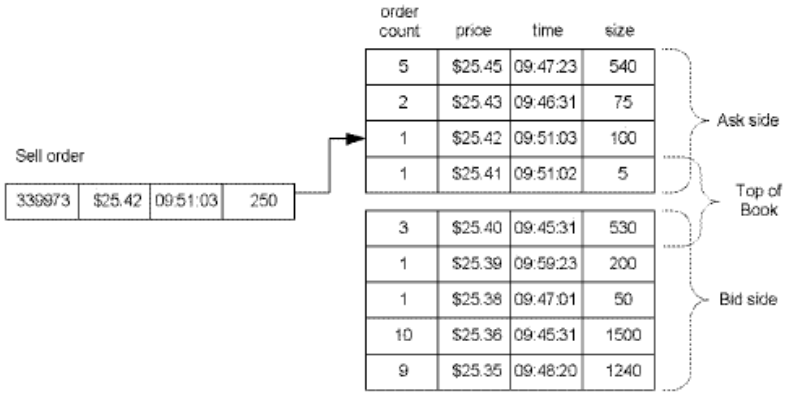

Figure 2(b)

Claim 43 does not require the claimed price point records include any particular fields. '115, 20:22-24 (price point records can "***have more fewer and/or different data fields***" that the examples shown); Donefer, ¶ 228. Thus, the

Modified-OBC's price-aggregated sub-records having price entries for each price point meet the claimed "*plurality of price point records*." Donefer, ¶ 228.

Additionally, although not required to meet the claimed price point records, a POSA would have understood that entries "***indicative*** of the total number of orders available at each price point" (Parsons, [0071]) would have included the aggregated number of shares in those orders, as the total number of shares available at each price point is the price-aggregation information of primary interest to traders. Donefer, ¶ 229. To the extent Parsons is not considered to expressly disclose this, the conventional and obvious way to implement the price-aggregated structures Parsons's OBC describes is to include the number of shares at each price point so that the disclosed price-aggregated views are most helpful to traders. Donefer, ¶ 229.

As with updating the order sub-records (*see* § IV.B.2.d above), to update the price-aggregated sub-records, the Modified-OBC uses the streaming limit order data to identify the corresponding financial instrument record. Parsons, [0093]-[0094]. The OBC's Memory Manager accesses the financial instrument record from the cache and the price-aggregated sub-records to be updated are placed into corresponding record buffers. Parsons, [0125]-[0132]; Donefer, ¶ 230; § IV.B.1.c.

The Modified-OBC's update engines assigned to perform updates to the price aggregated sub-records access those sub-records from the record buffer to

perform updates according to the information in the message stream. Parsons, [0125]-[0131]-[0132]; Donefer, ¶ 230; § IV.B.1.c.

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 are those whose fields are updated based on the order information (i.e., "*limit order event data*") extracted from the message stream. Parsons, [0071], [0096]; Donefer, ¶ 231.

g. [43D.2] "wherein the price engine is configured to … (2) compute updated price point data based on the accessed price point records and the streaming limit order event data"

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

Specifically, to ensure that order books accurately reflect up-to-date aggregated order information at each price point at which shares are available via outstanding limit orders, (1) new price-aggregate records are added when a limit order at a new price point is reported, (2) price-aggregate records are removed when a trade event (matching a bid with an ask) occurs for all outstanding shares at

a given price point; and (3) existing price-aggregated records are modified to reflect a new total share volume in response to a limit order at an existing price point. Donefer, ¶ 233.

Parsons's Modified-OBC is configured to perform all three of these types of update operations, and performing any one of them individually, and all of them collectively, meets the claimed "compute updated price point 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"); Donefer, ¶ 234.

A POSA would have understood that updating "the fields of the sub-records…in real-time with information contained in streaming messages," (Parsons, [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. Donefer, ¶ 235; Parsons, [0071], [0096]. To the extent Parsons is not considered to explicitly disclose this, that would have been the conventional and obvious way to maintain the price-aggregated records Parsons describes. Donefer, ¶¶ 236-39.

h. [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"

The '115 describes "enrich[ing]" streaming limit order events by modifying fields in, or appending fields to, limit order messages in the input message stream with the updates computed for the limit order and price point records. '115, 18:22-26; 18:61-63. Parsons's Modified-OBC discloses this same message enriching. Donefer, ¶ 240.

In Parsons, a trader may request a current market data view for a particular stock when contemplating a trade. Parsons, [0071], [0096], [0160]; Donefer, ¶¶ 246-247. That view becomes outdated as market conditions change, making speed of the essence for traders. *Id.;* Parsons, [0004]-[0005].

Parsons explains each of its caching modules (including the OBC and LVC) is able to provide a user with a view of its data (Parsons, [0060], [0062], [0071], [0096]), and to modify or augment limit order data (Parsons, [0095], [0100], [0125]) with updates from its message/record updater. Donefer, ¶¶ 243-245; § IV.B.1.c. Parsons provides an example of the LVC's message/record updater, which a POSA would have used in implementing the Modified-OBC. *See* § IV.B.1.c; Donefer, ¶¶ 158-170.

After a trader requests a view of a financial instrument, when the "***message***/record ***updater***" of the caching module supporting that view computes a

data update, it updates not only the cached records, but also the messages in the message stream. Parsons, [0125], [0100], [0132]; Donefer, ¶ 247. Specifically, when a message triggers a data update, the "message/record updater" modifies and/or augments the message with the computed updates in real-time, so that the trader's view of the market data stays current. *Id*; Parsons, [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].

Parsons does not describe the specific updates the OBC applies to the message stream, but discloses (using its LVC as an example) that its caching modules update messages in the message stream by modifying message fields or augmenting messages with additional fields based on other fields in the message or records maintained in the cache(s). Parsons, [0100], [0125], [0132]; Donefer, ¶ 248. A POSA would have understood Parsons to describe that, when updates are made by the OBC, a message/record updater modifies and/or augments the message in the message stream with the updates the OBC makes to the cached version of the order book. *Id*.

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, that would have been the obvious to way implement Parsons's OBC in view of Parsons's

discussion of the LVC message/record updater, and thus the obvious way to implement the Modified-OBC. Parsons, [0125]-[0133]; Donefer, ¶ 249.

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]; Donefer, ¶¶ 248-250. 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 RLDs are intended to do. Donefer, ¶¶ 241-242, 250.

Claim 43 requires enriching with "*financial market depth data*." The '115 uses "financial market depth data" to refer to messages processed to update order books ('115, 5:15-31) but does not describe its content. Donefer, ¶¶ 56-58.

The '115 defines "financial market data" ('115, 1:63-2:3), also uses it to describe messages processed to update order books (*id.*, 9:63-10:5, 12:28-34), and does not indicate that "financial market ***depth*** data" means anything different. Donefer, ¶¶ 251-253. 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, trades, etc. *Id.*; Parsons, [0071], [0096]; '115, 1:63-2:3.

Additionally, the "financial market depth data" the '115 enriches the streaming order limit events with is the updated order and price-aggregated information computed for the limit order event in the message stream, which is the same information Parsons's Modified-OBC computes. '115, 18:22-26; 18:61-63; 20:61-21:8; §§ IV.B.2.f-g; Donefer, ¶¶ 56-58, 254-256. Thus, "*financial market depth data*," covers the updated order and price-aggregated information computed by the Modified-OBC as discussed in connection with limitations [43C.2] and [43D.2] above. *Id.*; §§ IV.B.2.e, IV.B.2.g.

"[D]epth" data for a financial instrument is sometimes used to indicate the volume of shares available in the market. Donefer, ¶¶ 57, 257. 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 meets this narrower meaning. *Id.*

For the foregoing reasons, the Modified-OBS coprocessor meets [43E]. *Id.*, ¶ 258.

i. [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"

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

The order engine's "respective operations" are recited in limitations [43C.1] and [43C.2] and the price engine's are recited in limitations [43D.1] and [43D.2]. The Modified-OBC's order and price engines perform these operations in parallel as claimed. Donefer, ¶ 260.

As discussed in § IV.B.1.c, the Modified-OBC uses the update engine distribution scheme based on record type taught by Parsons, and thus distributes operations performed on its order sub-records across one or more "order" update engines (i.e., the "*order engine*"), and operations performed on its price-aggregated sub-records across one or more "price" update engines (i.e., the claimed "*price engine*,") as illustrated in modified Figure 15(a) below. Parsons, [0125], [0132], Figure 15(a); Donefer, ¶ 260.

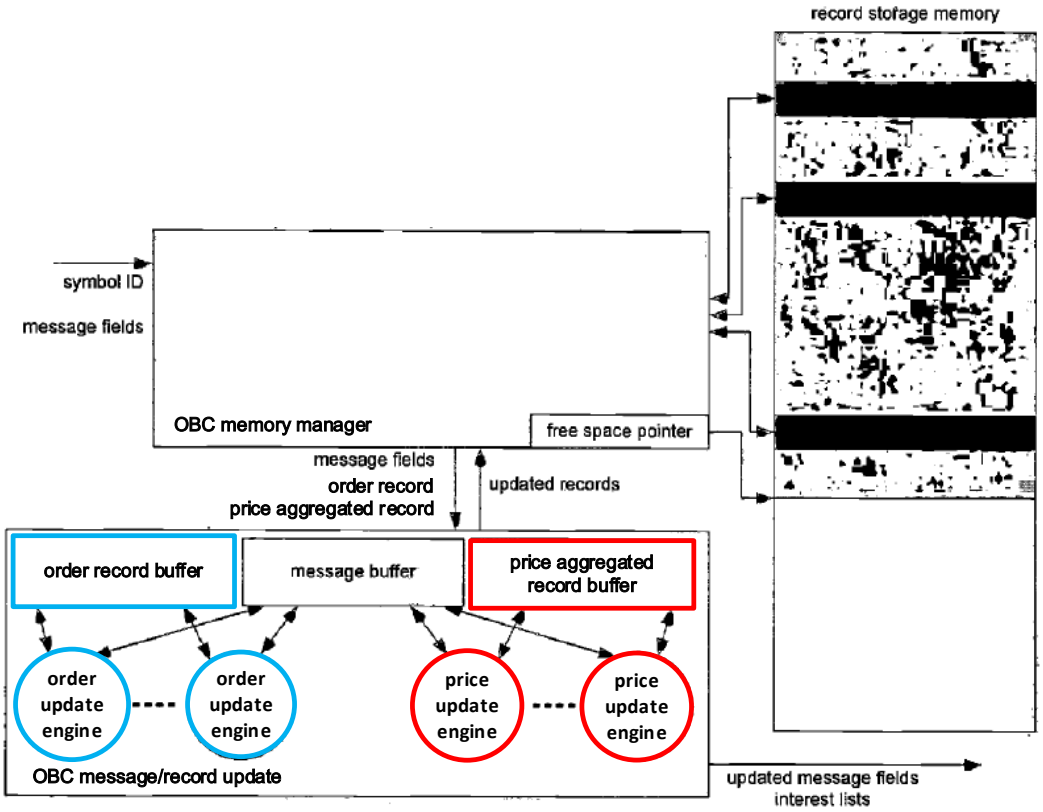

Figure 15(a)
(Modified)

As shown in § IV.B.1.c, the order and price-aggregated sub-records retrieved from the cache for updating based on the information extracted from the message stream (Parsons, [0093]-[0094], [0096], [0128]-[0131]) are loaded into their respective record buffers from which they are ***accessed*** and ***updated*** in parallel. *Id.*; Donefer, ¶ 261.

Claim 43 does not limit the "access" operations in limitation [43C.1] and [43D.1] to any particular manner of accessing. The Modified-OBC's order and price engines perform their respective "*access*" operations in parallel because they

access their corresponding record buffers in parallel to achieve the benefits Parsons describes. Donefer, ¶ 262; §§ IV.B.1.c, IV.B.2.d

Additionally, as discussed in §§ IV.B.2.e and IV.B.2.g, the updates to the order and price-aggregate sub-records performed by the Modified-OBC include computing the updated limit order and price-aggregated data recited in [43C.2] and [43D.2]. Donefer, ¶ 263. As explained in § IV.B.1.c, the Modified-OBC's order and price engines also perform their respective "*compute*" operations in parallel to achieve the performance benefits Parsons describes. Donefer, ¶ 263.

Thus, the Modified-OBC's order and price engines perform their respective "access" and "compute" operations in parallel and therefore meet limitation [43F]. Donefer, ¶ 263.

3. Claim 44

Claim 44 depends from claim 43 and requires the coprocessor be a member of a Markush group including "a reconfigurable logic device." The FPGA on which the Modified-OBS coprocessor is implemented (*see* § IV.B.1) is expressly characterized by Parsons as a reconfigurable logic device. Parsons, [0060], [0069]-[0070], [0163]-[0165], Figures 2-6; Donefer, ¶¶ 268-269. Meeting the "*reconfigurable logic device*" alternative recited in the Markush group is sufficient to satisfy this element. *Fresenius USA v. Baxter Int'l*, 582 F.3d 1288, 1298 (Fed. Cir. 2009); Donefer, ¶ 269.

Claim 44 further requires that the order and price engines be deployed on the member of the Markush group. As discussed in §§ IV.B.1.c., IV.B.2.b, the Modified-OBS processor includes the Modified-OBC so the order and price engines are deployed on the FPGA (the reconfigurable logic device).

C. Ground 1B: Claims 43-44 Are Rendered Obvious By An Obvious Implementation Of Parsons Where The OBS and LVC Are Deployed On The Same FPGA

1. Background to Ground 1B

a. Parsons's Market Platform

As discussed in § IV.A.2 *supra*, Parsons discloses a platform (600, Fig. 6) that consolidates a range of functionality (Parsons, [0014]) on RLDs, including device 604 on which the LVC server is implemented, and device 606 on which the OBS is implemented. Parsons, [0011]-[0015], [0056], [0060], [0070]. The LVC server on device 604 and the OBS on device 606 perform their operations on stream 106 in parallel to deliver "a variety of information on the financial markets with less latency." Parsons, [0056], [0162], Figure 6; Donefer, ¶¶ 271-275.

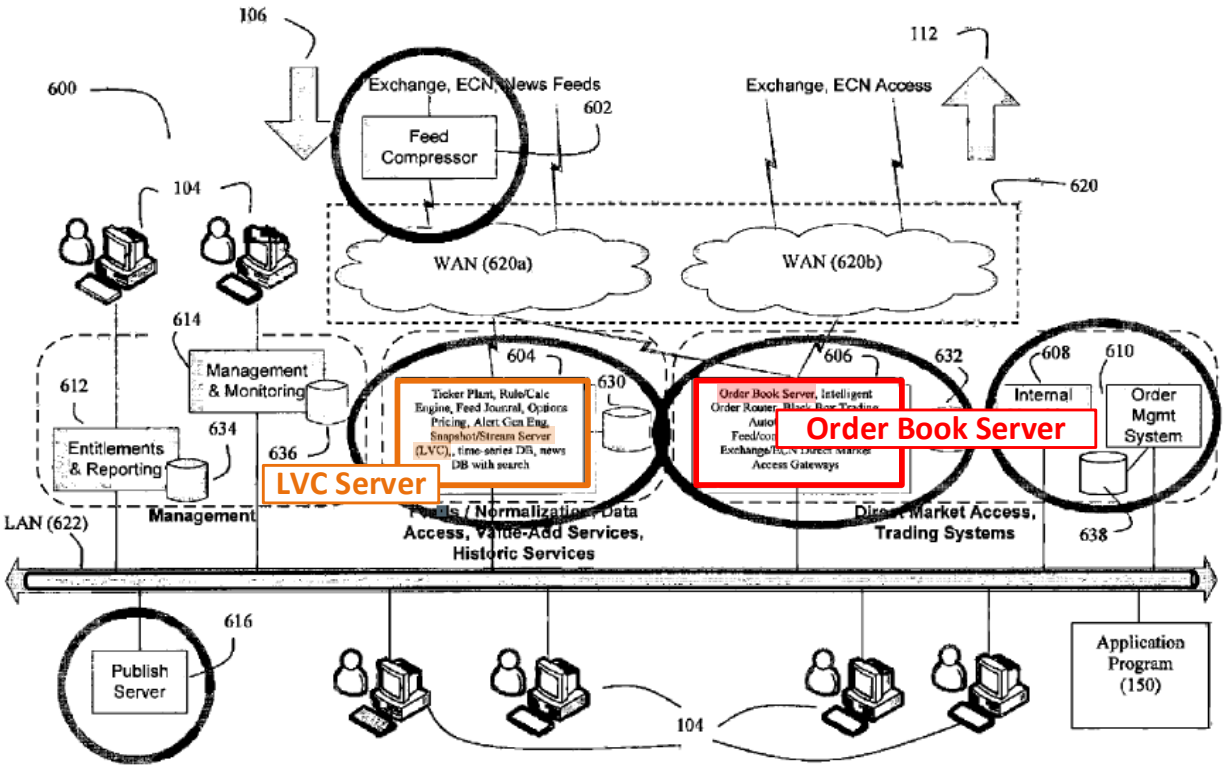

Figure 6

Parsons makes clear that Fig. 6's system 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." Parsons, [0165]; Donefer, ¶ 272.

Parsons describes various FAM pipelines to implement the functionality Parsons describes. Donefer, ¶ 273. Fig. 11 (below) illustrates one such exemplary FAM pipeline for providing "ticker plant" functionality. This "ticker plant" embodiment employs a price summary LVC caching module (Parsons, [0084], [0095]), illustrated as Value Cache Update (1106) and referred to as the Last Value Cache (LVC) Update FAM 1106 (hereinafter "LVC"). Donefer, ¶ 274. Like the

OBC, the LVC performs real-time updates to financial instrument records based on a message stream from market exchanges. *Id.*; Parsons, [0062], [0071].

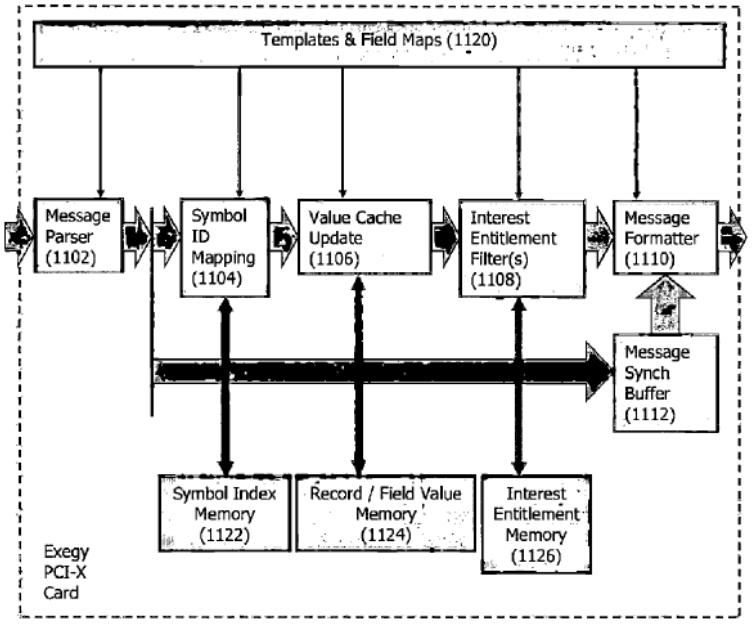

Figure 11

Parsons teaches that it is desirable to minimize the number of FPGAs to implement desired functionality. Parsons, [0011], [0056], [0165]; Donefer, ¶ 275. Indeed, Parsons suggests over time it may be possible to consolidate "***all*** functionality shown in Fig. 6 [including the OBS and LVC] on a single system 200" with a single FPGA. *Id.;* Parsons, [0165].

b. A POSA Would Have Been Motivated to Implement Parsons's OBS and LVC on a Single "Ticker Plant" FPGA

For the multiple reasons detailed below, a POSA would have been motivated to include Parsons's OBC in the ticker plant pipeline of FIG. 11 and consolidate the OBS and LVC on a single FPGA. Donefer, ¶¶ 276-287.

First, the OBS and the ticker plant embodiment of Fig. 11 process the same data stream, both provide time-sensitive financial market depth data to assist traders in understanding the state of the market, and they share certain FAM modules. Parsons, [0062], [0071], [0095]-[0096], [0126], [0162]. A POSA would have understood that integrating this functionality on a single FAM pipeline implemented on one FPGA would provide numerous efficiencies, including reduced hardware (a single FPGA), reduced processing redundancy (shared FAM modules), reduced software processing by the main processor in distributing the information output from the OBS and ticker plant to entitled subscribers, and increased simplicity for downstream components. Parsons, [0092], [0136]-[0138], [0162]; Donefer, ¶¶ 279, 283.

More specifically, by providing the financial market depth data provided by the OBC and 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, thereby reducing latency and simplifying the process of

delivering financial market depth data to subscribers interested in both types of enhancement. Donefer, ¶ 280. For example, below is an actual view for Microsoft stock traded on INET circa 2003 which includes both the type of financial market depth information provided by Parsons's OBC (highlighted red), and the type provided by Parsons's LVC (highlighted yellow). Parsons, [0071], [0096], [0126]; Donefer, ¶ 281.

In Parsons's platform 600 where the OBC and LVC are implemented on separate FPGAs, outputs from the two devices are routed to downstream consumers that subscribe to these services provided by the OBC and LVC. Parsons, [0133], [0136]-[0138]; Donefer, ¶ 282. By combining the OBC and LVC on the same pipeline, interest lists (identifying the subscribers) for the OBS and LVC are stored on the same FPGA and applied to a single output stream constructed by the Message Formatter 1110 (Fig. 11), simplifying the processing of the interest lists and the routing of updates to subscribers. Donefer, ¶ 282.

The same Message Parser, Symbol Mapping, Entitlement and Filtering and Message Formatter FAMs illustrated in FIG. 11 for the LVC are also described for use with the OBC and can be shared when the OBC and LVC are implemented on the same FPGA. Parsons, [0093]-[0094], [0096]; Donefer, ¶ 282.

Second, Parsons teaches combining functionality from its various FAM pipelines, explaining that some subset of the functions in FIG. 6 can be implemented on reconfigurable device 604 and a user can add additional functionality by reconfiguring the device "to add any desired new functionality." Parsons, [0165]; Donefer, ¶¶, 284-286; *see also* Parsons, [0063] ("numerous other FAM pipelines could be readily devised and developed by [POSAs] following the teachings herein").

Third, Parsons teaches that the OBS and LVC functionality is suitable to be combined on one FPGA, as Parsons envisioned consolidating "***all*** functionality" in FIG. 6 "on a single system 200." Parsons, [0165]; Donefer, ¶¶ 275, 287.

Fourth, Parsons claims a consolidated device comprising a "firmware application module pipeline" that is "configured to perform" one or more (i.e., "at least one") financial data processing operations from a set that includes both an "(LVC) server operation" and an "order book server [OBS] operation." Parsons, claim 24. Donefer, ¶ 287.

Fifth, 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. Donefer, ¶ 286; *KSR*, 550 U.S. *416.

c. Implementing the OBS on the Ticker Plant FAM Pipeline

Parsons's Fig. 11 ticker plant processes financial market data from 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. Parsons, [0126]; Donefer, ¶ 288.

A POSA seeking to consolidate this functionality with OBS functionality for the reasons detailed above would have added the OBC "hardware template" (Parsons, [0077]) onto the pipeline as taught by Parsons. Donefer, ¶ 289; Parsons, [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").

The same Message Parser, Symbol Mapping, Entitlement and Filtering FAMs illustrated in Fig. 11 for the LVC are used with the OBC (Parsons, [0093]-[0094], [0096]), so adding the OBC into the pipeline would have required minimal and straightforward changes. Donefer, ¶ 292; Parsons, [0044] ("FAMs can also be flexibly reprogrammed to change the parameters of their data processing operations"). The Message Formatter constructs an output stream by combining updated message streams from multiple upstream FAMs with the original message stream, and would have been the obvious choice to combine the updated message streams from the OBC and LVC. Parsons, [0108]; Donefer, ¶ 290.

It would have been well within the skill of a POSA to configure the FAM pipeline to add OBS functionality to Fig. 11's "ticker plant" by including the OBC in the FAM pipeline and reprogramming the FAM modules to the extent needed, as Parsons touts the simplicity of such programming. Parsons, [0044], [0068], [0077], [0083], [0163], [0165]; Donefer, ¶ 291.

As discussed in § IV.C.1.a above, the LVC and OBS processing is performed in parallel on separate devices to deliver low latency market information to traders. Parsons, [0011], [0056], [0162], FIG. 6; Donefer, ¶ 292. 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. *Id*.

Parsons teaches arranging FAMs in parallel in a pipeline to carry out a variety of data processing tasks. Parsons, [0106]-[0111], FIGS. 9, 10; Donefer, ¶ 293. One obvious way to implement a FAM pipeline to perform OBS and LVC functionality and maintain parallel operation would have been to arrange the OBC and LVC in parallel, as shown below in the Fig. 11 pipeline modified to include the OBC.

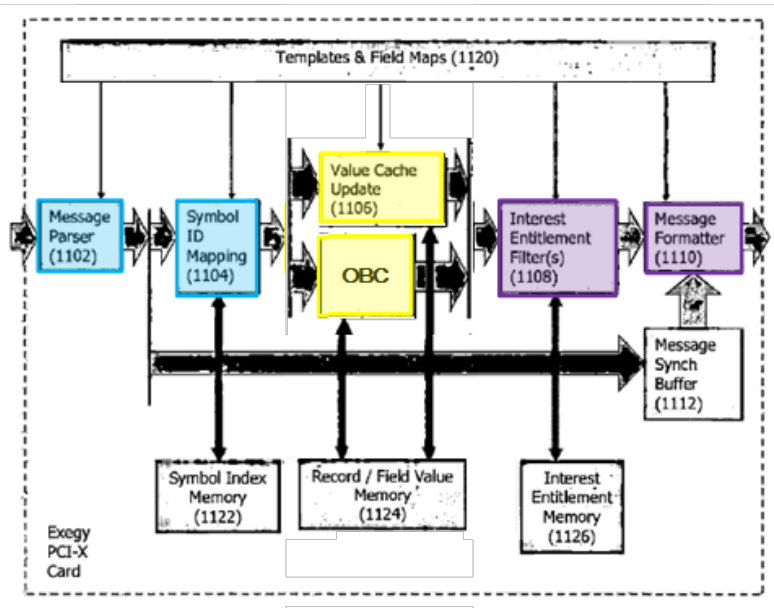

Figure 11
(Modified)

The above modified FAM pipeline is referred to herein as the Ticker Plant Server (TPS). The FPGA resource limitations Parsons mentions (Parsons, [0165]) would not have presented an obstacle to implementing the TPS since it does not combine ***all*** the Fig. 6 functionality onto a single device. Donefer, ¶ 294. The TPS combines only a subset of the functionality disclosed as being implemented on each of devices 604 and 606. Parsons, [0060], [0070]. FPGA technology in 2008 was fully capable of supporting the TPS on one FPGA. Donefer, ¶ 295. Indeed, the FPGA resources needed to support the TPS are less than those to support the suite of functionality in either of devices 604 and 606 of Fig. 6. *Id.*

2. Analysis of Challenged Claims

Several of the limitations are met by an FPGA implementing the TPS for the same reasons the Modified-OBS processor meets those limitations. Thus, for several limitations below cross-reference is made back to the analysis in Ground 1A, and that cross-referenced analysis is expressly incorporated into Ground 1B.

The headings below do not reproduce each limitation and instead identifies the limitation using the corresponding label (e.g., [43PRE], [43A], etc. from the attached Claims Listing.

3. An FPGA Implementing The TPS Renders Obvious Claim 43

a. [43PRE]

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. *See* § IV.B.2.a (a Parsons FPGA is "*specific computer technology*" that reduces latency and increases throughput with respect to streaming data enrichment); *see also* § IV.C.3.g below (demonstrating how the TPS coprocessor enriches the stream); Donefer, ¶¶ 171-174, 298.

b. [43A]

As discussed in § IV.C.1.c, the TPS FPGA is reconfigurable logic 202 implemented on board 400 (Parsons, [0060], [0069]), and is a "*coprocessor*" for the same reasons (*see* § IV.B.2.b) the FPGA implementing the Modified-OBS is.

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 in § IV.3 below, and are implemented (by a FAM on the FPGA) in the same manner as the "order engine" described in the '115. Donefer, ¶¶ 180-190, 300. Thus, the OBC includes an order engine for the same reasons the Modified-OBC does (*see* §§ IV.B.2.b.ii, IV.B.2.d-e *supra*), because the modifications made

to the OBC in Ground 1A (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. Donefer, ¶ 300.

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. Parsons, [0112], FIG. 11; Donefer, ¶ 301; §§ IV.B.2.b.ii, IV.C.1.a *supra*.

c. [43B]

Parsons's OBS pipeline (with the OBC) and ticker plant pipeline (with the LVC - Fig. 11) processes the same financial data stream 106 (Parsons, [0006], [0057], [0062], [0071], [0112]) processed by the Modified-OBS coprocessor in Ground 1A. The TPS coprocessor is configured to receive this stream and thus meets [43B] for the same reasons (*see* § IV.B.2.c) the coprocessor in Ground 1A does. Donefer, ¶ 302.

d. [43C.1]

The TPS implements Parsons's OBC, which meets both "*access*" and "*compute*" limitations in [43C] for at least the same reasons the Modified-OBC

does. *See* §§ IV.B.2.d-e. 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. Donefer, ¶¶ 303-309.

As discussed in §§ IV.B.2.d-e, 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. Donefer, ¶ 304. 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. Donefer, ¶ 304.

One obvious implementation for the OBC would have been to use the cache interface Parsons describes for other caching modules, including a Memory Manager to retrieve 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]; Donefer, ¶ 305; § IV.B.1.c. Unlike for Ground 1A, Ground 1B 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.

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 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]; Donefer, ¶ 306.

e. [43C.2]

The TPS implements Parsons's OBC, which meets [43C.2] for the same reasons the Modified-OBC does as discussed in § IV.A.2.e. Donefer, ¶¶ 303, 307-309; *see also* IV.C.3.d.

f. [43D.1]

As discussed § IV.B.2.f, the '115 does not define "*price point record*," and explains that price point records can "***have more fewer and/or different data fields***" than the examples described. '115, 20:22-24; Donefer, ¶ 310. Claim 43 does not limit the "*plurality of price point records*" to be limited to records reflecting outstanding orders or to have any specific field. Donefer, ¶ 310.

The OBC price engine discussed in Ground 1A maintains price-aggregated records for outstanding ***orders*** (*see* §§ IV.B.2.f-g), 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]; Donefer, ¶ 311. These composite and regional records include price point

data, including raw data on price points and other data derived therefrom. Parsons, [0126]-[0127]; Donefer, ¶ 311.

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. Donefer, ¶ 312. 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]; Donefer, ¶ 312. Thus, the LVC maintains a "*plurality of price point records*" as claimed. Donefer, ¶ 312.

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 § IV.B.2.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 § IV.B.2.d. *Id.*; Donefer, ¶ 313; § IV.C.1.c (illustrating TPS pipeline).

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 described in § IV.B.1.b above. Parsons, [0125]-[0138]; Donefer, ¶ 314. The price engine in the LVC (i.e., the "computational elements of the LVC that update the price point records" discussed in § IV.B.1.b) accesses the plurality of price point records from the record buffers and meets limitation [43D.1]. Donefer, ¶ 314.

g. [43D.2]

Parsons's LVC computes numerous updated price point data values that meet [43D.2]. For example, 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 reports a trade. Parsons, [0126]-[0127]; Donefer, ¶ 315. "*[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]; Donefer, ¶ 315.

In response to a reported trade event for a financial instrument, the LVC updates the accessed composite and regional records and the message fields according to logic programmed to update those records and messages based on the event type and financial instrument type. Parsons, [0125]-[0127]; Donefer, ¶ 316.

Numerous fields the LVC updates (Parsons, [0126]) involve computing updated values based on the information in the message stream and the information stored in the fields of the accessed records, including ***each*** of the above-mentioned "derived fields." Donefer, ¶¶ 317-318; *see also* Parsons, [0124] (describing computing a new tick direction by determining "if the price in the message is larger or smaller than the previous price captured in the record.").

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*." Donefer, ¶ 319.

h. [43E]

The TPS coprocessor meets [43E] two independent ways.

First, 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 in § IV.B.2.h *supra*, because the modifications to Parsons's OBC in Ground 1A do not impact the manner in which Parsons's OBC enriches the stream. Donefer, ¶¶ 320-321.

Second, the TPS coprocessor includes the LVC which also updates streaming order events in the message stream with updated price summary information, including (among other derived price point values) updated price-aggregated trade information as discussed above. *Id.;* Parsons, [0060], [0062], [0095], [0127]-[0133]. Parsons does not disclose that every LVC-computed update 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 maintained in the financial instrument records (Parsons, [0126]) and with which the message stream is updated. *Id.*, [0124]; Donefer, ¶ 323.

A POSA would have understood from this disclosure that the LVC updates limit order event messages in the message stream with 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/or 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 § IV.C.3.f-IV.C.3.g *supra*. Donefer, ¶¶ 323-324.

Each of the above-discussed price point records stored in the composite and regional records and updated in real-time are financial market depth data, as each represents some aspect of the state of the market for the corresponding financial instrument. *Id.*; Parsons, [0062] ("Each [LVC] 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 [43E].

Moreover, the LVC maintains and updates "total trade volume" at bid and ask. Parsons, [0126]. This meets the narrowest possible meaning of "financial market depth data" (*see* § IV.B.2.h above), and a POSA would have understood the LVC to enrich limit order events in the stream with this information in the same manner discussed above. Parsons, [0100], [0132]; Donefer, ¶ 324 (explaining total trade volumes meet the '115's definition of "financial market data" because they are data derived from messages representing a completed sale).

Thus, the TPS coprocessor meets [43E] for this additional reason. Donefer, ¶ 325.

Additionally, the TPS pipeline's Message Formatter (Fig. 11) receives the updated message streams from the OBC and LVC and constructs an output message by applying the OBC and LVC updates to the original message stream. § IV.C.1.c; Parsons, [0108], [0138]; Donefer, ¶ 326. This is accomplished by appending message fields from the OBC and LVC to the original message, replacing message fields with those from the OBC and LVC, or a combination. *Id*.

Thus, the TPS coprocessor meets [43E] for each of these reasons. Donefer, ¶ 320-321, 325-327.

i. [43F]

As explained in § IV.C.1.c and illustrated below, in the TPS coprocessor the OBC and LVC (*i.e.*, the claimed "*order engine*" and "*price engine*," respectively) are arranged as parallel FAMs in the TPS pipeline and thus perform their respective operations in parallel. Donefer, ¶¶ 328-329.

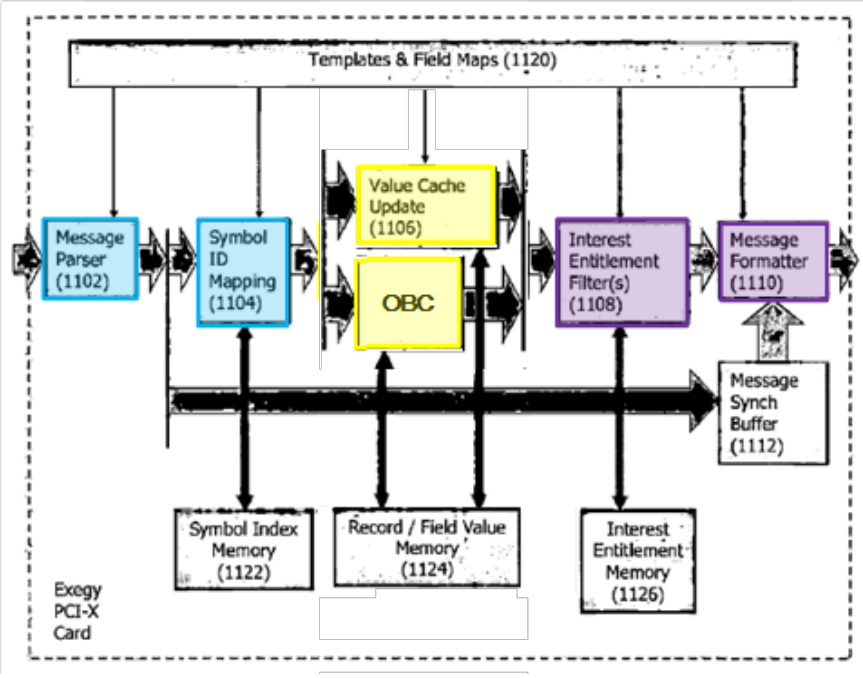

Figure 11
(Modified)

The update operations performed by the OBC and LVC do not depend on each other and are amenable to this parallel arrangement. Donefer, ¶ 330. In fact, arranging the OBC and LVC in series would have reduced throughput and increased latency, contrary to Parsons's teachings. *Id.*

In the parallel arrangement used in the TPS coprocessor, Message Parser 1102 parses limit order events reported in stream 106 from exchanges common to both caching modules and delivers them to the OBC and LVC for parallel processing. Donefer, ¶ 330. The "respective operations" of the order and price engines recited in claim 43 are those in [43C] and [43D]-[43D2] that relate to accessing the engine's respective records and computing updates to them.

Donefer, ¶ 331. In the TPS coprocessor the computing operations performed by the OBC's order engine and the TPC's price engine are done in parallel. *Id.* Similarly, in one obvious implementation where the OBC uses a Memory Manager and record buffers (*see* § IV.B.1.c), 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]. Donefer, ¶ 331.

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. Donefer, ¶ 332.

With respect to how the records are accessed from cache in the TPS coprocessor, 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]; Donefer, ¶ 332. In this way, memory accesses to the records maintained by the order and price engines are also performed in parallel.

Id. Thus, even if [43F] required that accesses to cache occur in parallel (it does not) the TPS coprocessor would meet [43F]. *Id*.

4. Claim 44

The TPS coprocessor meets claim 44 because it is implemented on an FPGA, which is a "*reconfigurable logic device"* that meets claim 44's Markush group for the same reasons discussed in § IV.B.3 for Ground 1A, and the TPS's order and price engines are both "deployed" on the FPGA. *See* § IV.C.1.c; Donefer, ¶¶ 332-35.

D. Claims 43-44 Are Rendered Obvious Even If Recited Elements Are Considered to be § 112, ¶6 Elements

If Patent Owner were to argue that any of the claimed coprocessor, order engine or price engine should be interpreted under 35 U.S.C. § 112, ¶ 6, the Modified-OBS coprocessor and TPS coprocessor still meet the claimed coprocessor, and the Modified-OBC and TPS still include the claimed engines. Donefer, ¶¶ 265-67. Petitioner does not believe these terms should be construed under § 112, ¶ 6. They lack the word "means" and are presumed not to be interpreted under § 112, ¶ 6. *Williamson v. Citrix Online, LLC*, 792 F.3d 1139, 1349 (Fed. Cir. 2015). Additionally, "coprocessor" and "engine" are known elements in the computing art. Donefer, ¶ 265.

If any of these terms were interpreted under § 112, ¶ 6, they are still met by the Modified-OBS coprocessor and the TPS coprocessor. *Id.* The functions

recited in claim 43 are all met by the Modified-OBS coprocessor and TPS coprocessor in the manner detailed in § IV.B and § IV.C, respectively. The Parsons structures in Grounds 1A and 1B that perform those functions are the same (or equivalent) structures the '115 describes. Donefer, ¶ 265.

Specifically, the Modified-OBS coprocessor (*see* § IV.B.2.a) and TPS coprocessor (*see* § IV.C.3.a) each is an FPGA programmed with FAMs, which is the same coprocessor structure in the '115. '115, 9:54-12:11, Figures 8-9; Donefer, ¶ 266. With respect to the order and price engines, the structure the '115 specification discloses for implementing each is a firmware application module (FAM) that programs the coprocessor (e.g., FPGA). Donefer, ¶ 267; '115, 10:26-12:34. The order and price engines in the Modified-OBS coprocessor (*see* §§ IV.B.1 and IV.B.2.b), and the order and price engines in the TPS coprocessor (*see* §§ IV.C.1 and IV.C.3.b) each is implemented the same way, i.e., on a FAM that programs an FPGA. Donefer, ¶ 265.

Additionally, Parsons is incorporated by reference into the '115 ('115, 1:20-24 and 1:32-33) and its disclosure forms part of the '115's supporting disclosure for the order and price engines.

V. THE BOARD SHOULD INSTITUTE TRIAL

A. No Basis Exists To Deny Institution Under § 314(a)

Discretionary denial is unwarranted under 35 U.S.C. § 314(a) because the parallel litigation is not at an "advanced state." *Apple v. Fintiv*, IPR2020-00019, Paper 11, 2 (PTAB Mar. 20, 2020) (precedential) (quoting *NHK Spring Co. v. Intri-Plex Techs.*, IPR2018-00752, Paper 8, 20 (PTAB Sep. 12, 2018) (precedential)). The complaint is subject to a pending motion to dismiss, no answer has been filed, the parties have engaged in little discovery, claim construction does not begin until November 6, 2020, with a *Markman* hearing scheduled for February 5, ***2021***, and no trial date has been set. Ex. 1012 at 1, 3; *Uniden Am. v. Escort*, IPR2019-00724, Paper 6, 4-10 (PTAB Sept. 17, 2019).

B. No Basis Exists To Deny Institution Under § 325(d)

The Board uses a two-part framework to assess whether discretionary denial under § 325(d) is appropriate: "(1) whether the same or substantially the same art…or…arguments previously were presented to the Office; and (2) if [so]…, whether the petitioner has demonstrated that the Office erred in a manner material to the patentability of challenged claims." *Advanced Bionics v. Med-El Elektromedizinsche Gerate*, IPR2019-01469, Paper 6, 8 (PTAB Feb. 13, 2020) (precedential). Non-limiting factors (a)-(b) and (d) articulated in *Becton Dickinson & Co. v. B. Braun Melsungen AG*, IPR2017-01586, Paper 8, 17-18 (PTAB Dec.

15, 2017) (informative) apply to the first part of the framework; factors (c) and (e)-(f) apply to the second part. *Advanced Bionics*, IPR2019-01469, Paper 6, at 8.

As Parsons was cited during prosecution, the first part of *Advanced Bionics'* framework is satisfied, but the second part is not because Ground 1 raises ***materially different*** arguments than those the Office previously considered, and the Examiner ***demonstrably failed to appreciate*** that the obvious implementations of Parsons relied upon herein meet the challenged claims.

1. *Becton, Dickinson* Factor (c): the Petition's Art and Arguments Were Previously Not Evaluated for Claims 43-44

First, the Examiner never applied Parsons in rejecting claims 43-44, which prior to amendments were original claims 42-43. Ex. 1002, 66-67 (original claims). The Examiner rejected original claims 1-2 and 41-43 as obvious over "Hamati" and "Kelleher." *Id.*, 1540. While ***other*** claims were rejected in view of Parsons (in combination with Hamati and Kelleher), the challenged claims were not. *Id*.

Second, the Examiner cited Parsons only for limitations (e.g., "summary view" and "ticker plant") not in claims 43-44. *Id.*, 1541-1548.

Third, Parsons materially differs from the Hamati-Kelleher combination. Hamati discloses processing streamed limit order event data to reconstitute the order book and generating a stream of enriched limit order events. *Id.*, 1540.

Kelleher discloses a graphics processing unit ("GPU"). *Id.* Neither Hamati nor Kelleher discloses ***how*** a GPU could be used to process streaming financial data. *Id.*, 1729. In contrast, Parsons discloses an FPGA coprocessor and obvious implementations of what Parsons discloses meet all of the elements of claims 43-44. Donefer, ¶¶ 142-149. Parsons is a lengthy reference that does not use the same "order engine" and "price engine" terms claims 43-44 do, and there is no indication the Examiner even recognized that Parsons discloses such engines, let alone considered combining Parsons's various teachings in the manner relied upon in Ground 1A or 1B.

Thus, there is ***no overlap*** between the arguments considered during prosecution and how Petitioner relies on Parsons. *Trans Ova Genetics v. XY*, IPR2018-00250, Paper 9, 18 (PTAB June 27, 2018) (granting institution where the "Petition presents a substantially different argument than was articulated by the Examiner."); *Asetek Danmark A/S v. Coolit Systems*, IPR2019-00705, Paper 19, 9-11 (PTAB Sept. 6, 2019) (granting institution where "the same art may have been considered during examination, [but] the Examiner did not evaluate Petitioner's [arguments]."); *Apple v. MPH Technologies Oy*, IPR2019-00819, Paper 10, 14-15 (PTAB Sept. 27, 2019) (granting institution where "there is little, if any, overlap between the arguments made during examination and the manner in which Petitioner now relies on [the cited art]."); *Puma North America v. Nike*, IPR2019-

01058, Paper 10, 18 (PTAB Oct. 31, 2019) ("Because…the manner in which Petitioner relies on the prior art does not overlap with arguments made during prosecution….this factor weighs against" discretionary denial).

2. *Becton, Dickinson* Factors (e)-(f): the Examiner Failed to Appreciate Parsons' Full Scope

The Examiner materially erred by failing to appreciate the full scope of Parsons's teachings relied upon herein. After original claims 42-43 were rejected over Hamati-Kelleher, the claims were amended to recite, *inter alia*, that the claimed coprocessor "includes an order engine and a price engine" that "perform their steps in parallel." Ex. 1002, 1694-95. Applicants argued Hamati and Kelleher "are silent about distributing tasks between an order engine and price engine in parallel." *Id.*, 1707.

In allowing the claims (*id.*, 1729), the Examiner failed to appreciate that Parsons's collective teachings relied upon herein render obvious (in two different ways) a coprocessor with order and price engines that perform their tasks in parallel, likely because ***Parsons does not use the terms "order engine" and "price engine."*** *Supra* §§ IV.B.2.i, IV.C.2.a.viii. The Examiner thus failed to appreciate Parsons's full disclosure.

Where the Examiner failed to appreciate the full scope of the prior art, the Board has instituted. *Apple v. Omni Medsci*, IPR2020-00029, Paper 7, 55 (PTAB April 22, 2020) ("the Office erred in a manner material to patentability in its

treatment of the art by failing to reject the claims…over the references cited in Petitioner's challenges."); *Trans Ova Genetics*, IPR2018-00250, Paper 9, 18-19 ("the Examiner manifestly failed to appreciate [disclosures] …from [the cited art]"); *Apple*, IPR2019-00819, Paper 10, 15-16 ("the Examiner did not properly apply [the prior art's] teachings").

The Petition is supported by a new expert declaration (Donefer, ¶¶ 3-21), explaining how a POSA would have understood Parsons (*id.*, ¶¶ 97-141), including that it discloses order and price engines despite not using that terminology. This new evidence weighs against discretionary denial. *Puma*, IPR2019-01058, Paper 10, 19 (instituting where petition presented "new non-cumulative evidence…. probative to issues of patentability and helpful to our consideration of a prior art combination that was not before the Examiner"); *Apple*, IPR2019-00819, Paper 10, 16-17 (similar).

3. Conclusion

Becton Dickinson factors (c) and (e)-(f) weigh in favor of institution. The Examiner never applied Parsons against the challenged claims (factor (c)) and misunderstood Parsons's full scope (factor (e)). The Petition also provides new expert evidence to better understand Parsons's disclosure (factor (f)).

VI. CONCLUSION

Cancellation of claims 43-44 is requested.

Dated: May 29, 2020

Respectfully submitted,

ACTIV Financial Systems, Inc.

By /Andrew J. Tibbetts/

Richard F. Giunta, Reg. No. 36,149
Andrew J. Tibbetts, Reg. No. 65,139
Marc S. Johannes, Reg. No. 64,978
Michael N. Rader, Reg. No. 52,146
WOLF GREENFIELD & SACKS, P.C.
600 Atlantic Ave.
Boston, MA 02210-2206
Tel: 617-646-8000/Fax: 617-646-8646

CLAIM LISTING

The following claim listing assigns element labels (e.g., [43PRE], [43A], etc.) to certain claims for clarity.

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
[43C.1] (1) access a plurality of limit order records based on the streaming limit order event data, and
[43C.2] (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
[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 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.

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.

CERTIFICATE OF SERVICE UNDER 37 C.F.R. § 42.6 (e)(4)

Pursuant to Pursuant to 37 C.F.R. §§ 42.6 (e) and 42.105(b), the parties agreed to electronic service (via email) of the Petition and all associated exhibits and papers. The undersigned certifies that, on May 29, 2020, I will cause a copy of the foregoing document to be served on Patent Owner via electronic mail at the e-mail addresses provided below.

J. Bradley Luchsinger	bluchsinger@HDP.com
Matthew Cutler	mcutler@HDP.com
Glenn Forbis	gforbis@HDP.com
Michael Thomas	mthomas@HDP.com

The undersigned also certifies courtesy copies are being sent to the attorney of record for the patent owner via Express Mail at the following address:

Thompson Coburn LLP
One US Bank Plaza
Suite 3500
St. Louis, MO 63101

Date: May 29, 2020

/Alexandra H. Kime/
Alexandra H. Kime
Paralegal
WOLF GREENFIELD & SACKS, P.C.

CERTIFICATE OF WORD COUNT

Pursuant to 37 C.F.R. § 42.24, the undersigned certifies that the foregoing Petition for *Inter Partes* Review contains 13,995 words excluding a table of contents, a table of authorities, Mandatory Notices under § 42.8, a certificate of service or word count, or appendix of exhibits or claim listing. Petitioner has relied on the word count feature of the word processing system used to create this paper in making this certification.

Date: May 29, 2020

/Alexandra H. Kime/
Alexandra H. Kime
Paralegal
WOLF GREENFIELD & SACKS, P.C.