



PCT

(10) International Publication Number
WO 2009/089467 A2

(51) International Patent Classification:

G06Q 40/00 (2006.01) **G06F 17/10** (2006.01)
G06F 17/40 (2006.01)

(21) International Application Number:

PCT/US2009/030623

(22) International Filing Date: 9 January 2009 (09.01.2009)

(25) **Filing Language:** English

(26) **Publication Language:** English

(30) Priority Data:

12/013,302	11 January 2008 (11.01.2008)	US
------------	------------------------------	----

(71) Applicant (for all designated States except US): EXEGY INCORPORATED [US/US]; 349 Marshall Avenue, Suite 100, St. Louis, MO 63119 (US).

(72) Inventors; and

(75) **Inventors/Applicants (for US only):** TAYLOR, David, E.

[US/US]; 3448 Missouri Avenue, St. Louis, MO 63118 (US). **SINGLA, Naveen** [IN/US]; 530 Union Boulevard, Apt. 605, St. Louis, MO 63108 (US). **BRODIE, Benjamin, C.** [US/US]; 363 Mount Royall Drive, Mount Pleasant, SC 29464 (US). **MCVICAR, Nathaniel, S.** [US/US]; 4615 Lindell Boulevard, Apt. 705, St. Louis, MO 63108 (US). **THIEL, Justin, R.** [US/US]; 14 Cottonwood Glen Drive, Glen Carbon, IL 62034 (US). **INDECK, Ronald, S.** [US/US]; 729 Gralee Lane, St. Louis, MO 63132 (US).

(74) **Agent:** VOLK, Benjamin, L.; Thompson Coburn LLP,
One Us Bank Plaza, St. Louis, MO 63101 (US).

(81) Designated States (*unless otherwise indicated, for every kind of national protection available*): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TJ,

[Continued on next page]

(54) Title: METHOD AND SYSTEM FOR LOW LATENCY BASKET CALCULATION

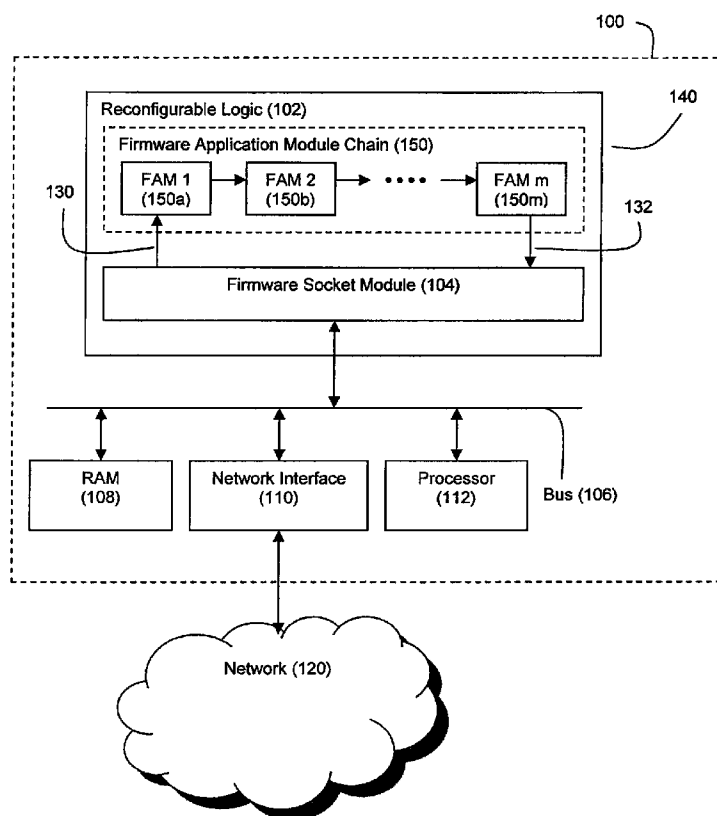


Figure 1(a)

(57) Abstract: A basket calculation engine is deployed to receive a stream of data and accelerate the computation of basket values based on that data. In a preferred embodiment, the basket calculation engine is used to process financial market data to compute the net asset values (NAVs) of financial instrument baskets. The basket calculation engine can be deployed on a coprocessor and can also be realized via a pipeline, the pipeline preferably comprising a basket association lookup module and a basket value updating module. The coprocessor is preferably a reconfigurable logic device such as a field programmable gate array (FPGA).



TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

(84) Designated States (*unless otherwise indicated, for every kind of regional protection available*): ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI,

Published:

— *without international search report and to be republished upon receipt of that report*

Method and System for Low Latency Basket Calculation

5 Field of the Invention:

 The present invention relates to the field of performing basket calculation operations based on streaming data, particularly streaming financial market data.

10 Terminology:

 The following paragraphs provide several definitions for various terms used herein. These paragraphs also provide background information relating to these terms.

15 Financial Instrument: As used herein, a "financial instrument" refers to a contract representing an equity ownership, debt, or credit, typically in relation to a corporate or governmental entity, wherein the contract is saleable. Examples of financial instruments include stocks, bonds, commodities, currency traded on currency
20 markets, etc. but would not include cash or checks in the sense of how those items are used outside the financial trading markets (i.e., the purchase of groceries at a grocery store using cash or check would not be covered by the term "financial instrument" as used herein; similarly, the withdrawal of \$100 in cash from an
25 Automatic Teller Machine using a debit card would not be covered by the term "financial instrument" as used herein).

 Financial Market Data: As used herein, the term "financial market data" refers to data contained in or derived from a series of
30 messages that individually represent a new offer to buy or sell a financial instrument, an indication of a completed sale of a financial instrument, notifications of corrections to previously-reported sales of a financial instrument, administrative messages related to such transactions, and the like.

35

Basket: As used herein, the term "basket" refers to a collection comprising a plurality of elements, each element having one or more values. The collection may be assigned one or more Net Values (NVs), wherein a NV is derived from the values of the plurality of elements in the collection. For example, a basket may be a collection of data points from various scientific experiments. Each data point may have associated values such as size, mass, etc. One may derive a size NV by computing a weighted sum of the sizes, a mass NV by computing a weighted sum of the masses, etc. Another example of a basket would be a collection of financial instruments, as explained below.

Financial Instrument Basket: As used herein, the term "financial instrument basket" refers to a basket whose elements comprise financial instruments. The financial instrument basket may be assigned one or more Net Asset Values (NAVs), wherein a NAV is derived from the values of the elements in the basket. Examples of financial instruments that may be included in baskets are securities (stocks), bonds, options, mutual funds, exchange-traded funds, etc. Financial instrument baskets may represent standard indexes, exchange-traded funds (ETFs), mutual funds, personal portfolios, etc. One may derive a last sale NAV by computing a weighted sum of the last sale prices for each of the financial instruments in the basket, a bid NAV by computing a weighted sum of the current best bid prices for each of the financial instruments in the basket, etc.

GPP: As used herein, the term "general-purpose processor" (or GPP) refers to a hardware device having a fixed form and whose functionality is variable, wherein this variable functionality is defined by fetching instructions and executing those instructions, of which a conventional central processing unit (CPU) is a common example. Exemplary embodiments of GPPs include an Intel Xeon processor and an AMD Opteron processor.

Reconfigurable Logic: As used herein, the term "reconfigurable logic" refers to any logic technology whose form and function can be

significantly altered (i.e., reconfigured) in the field post-manufacture. This is to be contrasted with a GPP, whose function can change post-manufacture, but whose form is fixed at manufacture.

- 5 Software: As used herein, the term "software" refers to data processing functionality that is deployed on a GPP or other processing devices, wherein software cannot be used to change or define the form of the device on which it is loaded.
- 10 Firmware: As used herein, the term "firmware" refers to data processing functionality that is deployed on reconfigurable logic or other processing devices, wherein firmware may be used to change or define the form of the device on which it is loaded.
- 15 Coproprocessor: As used herein, the term "coprocessor" refers to a computational engine designed to operate in conjunction with other components in a computational system having a main processor (wherein the main processor itself may comprise multiple processors such as in a multi-core processor architecture). Typically, a
- 20 coprocessor is optimized to perform a specific set of tasks and is used to offload tasks from a main processor (which is typically a GPP) in order to optimize system performance. The scope of tasks performed by a coprocessor may be fixed or variable, depending on the architecture of the coprocessor. Examples of fixed coprocessor
- 25 architectures include Graphics Processor Units which perform a broad spectrum of tasks and floating point numeric coprocessors which perform a relatively narrow set of tasks. Examples of reconfigurable coprocessor architectures include reconfigurable logic devices such as Field Programmable Gate Arrays (FPGAs) which
- 30 may be reconfigured to implement a wide variety of fixed or programmable computational engines. The functionality of a coprocessor may be defined via software and/or firmware.

Hardware Acceleration: As used herein, the term "hardware

35 acceleration" refers to the use of software and/or firmware implemented on a coprocessor for offloading one or more processing

tasks from a main processor to decrease processing latency for those tasks relative to the main processor.

5 Bus: As used herein, the term "bus" refers to a logical bus which encompasses any physical interconnect for which devices and locations are accessed by an address. Examples of buses that could be used in the practice of the present invention include, but are not limited to the PCI family of buses (e.g., PCI-X and PCI-Express) and HyperTransport buses.

10 Pipelining: As used herein, the terms "pipeline", "pipelined sequence", or "chain" refer to an arrangement of application modules wherein the output of one application module is connected to the input of the next application module in the sequence. This
15 pipelining arrangement allows each application module to independently operate on any data it receives during a given clock cycle and then pass its output to the next downstream application module in the sequence during another clock cycle.

20 Background and Summary of the Invention:

The ability to closely track the value of financial instrument baskets throughout a trading day as large volumes of events, such as trades and quotes, are constantly occurring is a daunting task. Every trade made on a financial instrument which is part of the
25 portfolio of financial instruments underlying a financial instrument basket will potentially cause a change in that basket's value. The inventors herein believe that conventional techniques used to compute basket values have been unable to keep up with the high volume of events affecting basket values, thereby leading to
30 inaccuracy for financial instrument basket values relative to the current state of the market because, with conventional techniques, the currently computed basket value will not be reflective of current market conditions, but rather market conditions as they existed anywhere from milliseconds to several seconds in the past.

35 As noted above, financial instrument baskets may represent standard indexes, ETFs, mutual funds, personal portfolios, etc.

An index represents the performance of a group of companies, wherein the group can be decided by various criteria such as market capitalization (e.g., the S&P 500), industry (e.g., the Dow Jones Industrial Average (DJIA)), etc.

5 A mutual fund is a professionally-managed form of collective investment wherein the buyer pays the fund owner a certain amount of money and in exchange receives shares of the fund.

ETFs are very similar to mutual funds with an important difference being that ETFs can be traded continuously throughout the trading day on an exchange, just like stocks. Therefore, the prices of ETFs fluctuate not only according to the changes in their underlying portfolios, but also due to changes in market supply and demand for the ETFs' shares themselves. Thus, ETFs provide the trading dynamics of stocks as well as the diversity of mutual funds. Additionally, ETFs typically track a well-established market index, trying to replicate the returns of the index.

Personal portfolios are groupings of financial instruments that are defined by an individual for his/her personal investment goals.

20 A financial instrument basket is defined by its set of underlying financial instruments. The basket definition also specifies a weight for each of the financial instruments within the basket. For example, consider a hypothetical basket denoted by B . Basket B contains M financial instruments with symbols C_i , wherein $i=1, 2, \dots, M$. Further, the weights of each financial instrument i in the basket is denoted by w_i , wherein $i=1, 2, \dots, M$. Associated with a basket is a measure of its monetary value known as its net asset value (NAV). If one assumes that the value at the current instant of each financial instrument i within basket B is T_i , wherein $i=1, 2, \dots, M$, then the NAV of basket B can be computed as:

$$NAV = \frac{1}{d} \sum_{i=1}^M w_i T_i \quad (1)$$

wherein d is a divisor that is associated with the basket and discussed in greater detail hereinafter. Thus, equation (1) specifies that a basket's NAV is the weighted sum of the worth of

the financial instruments within the baskets, wherein the divisor acts as a normalizing factor.

The value of T represents a price for the financial instrument's shares. The price can be expressed as the last sale/trade price (last), the best offer to buy price (bid), the best offer to sell price (ask), or specific offer prices to buy or sell (limit orders).

The value of w can be defined differently for each basket. For example, with capitalization-weighted indexes (such as the S&P 500), the weight of the financial instrument can be defined as the number of outstanding shares of the financial instrument multiplied by a "free-float" factor, wherein the free-float factor is a number between 0 and 1 that adjusts for the fact that certain shares of the financial instrument are not publicly available for trading. With an equal-weighted index (such as the DJIA), the weight of a financial instrument is set to 1. With ETFs and personal portfolios, the weight of a financial instrument can be set equal to a specified number which reflects the fact that owning many shares for each of the basket's financial instruments amounts to owning one share of that ETF or personal portfolio. Thus, in order to own one share of basket B , one would have to purchase w_1 shares of financial instrument C_1 , w_2 shares of financial instrument C_2 , and so on up to w_M shares of financial instrument C_M .

The value of d can also be defined differently for each basket. For an index, the divisor can be set to the current index divisor, which can be either the number of financial instruments in the index (e.g., 500 for the S&P 500) or some other specified value. For ETFs, the divisor can change during the trading day (as the divisor can also do with respect to indexes).

As indicated above, whenever there is a change in worth for one of a basket's underlying financial instruments, then the NAV for that basket should be calculated in accordance with formula (1). However, financial market data messages from exchanges stream into a platform such as a ticker plant at very high rates. For example, the options price reporting authority (OPRA) expects that its message rate in January 2008 will be approximately 800,000

messages/second. Each of these messages may potentially trigger multiple NAV recalculations. Moreover, because the same financial instrument may be a member of several baskets, the inventors herein estimate that, in order to track basket NAV values for all financial market data messages, the number of NAV calculations that will need to be performed every second will be at least on the order of 10 million NAV calculations/second. The inventors herein believe that conventional solutions are unable to keep up with such high update rates due to the serial nature of the conventional solutions, which constrain it to perform only one update at a time (including other tasks that a system may need to perform unrelated to basket calculations).

This shortcoming is problematic because it is highly desirable for financial traders to be able to calculate the basket NAVs extremely fast. An example will illustrate this desirability. As explained above, ETFs are baskets which can be traded like regular stocks on an exchange. The portfolio for an ETF is defined at the beginning of the trading day and remains fixed throughout the trading day barring rare events. Shares of ETFs can be created throughout the trading day by a creation process which works as follows. A buyer gives the owner of a fund the specified portfolio of financial instruments that make up the ETF and a cash payment in an amount equal to the dividend payments on those financial instruments (and possibly a transaction fee). The fund then issues shares of the ETF to the buyer. Shares of ETFs can also be redeemed for the underlying financial instruments in a similar manner: the investor gives the fund owner shares of the ETF (and possibly some cash as a transaction fee), and in return the investor receives the portfolio of financial instruments that make up the ETF and cash equal to the dividend payments on those financial instruments.

During the trading day, the price of the shares of the ETF fluctuates according to supply and demand, and the ETF's NAV changes according to the supply and demand of the financial instruments in the ETF's portfolio. It may well happen that, during the trading day, the trading price of the ETF deviates from its NAV. If the price of the ETF is higher than the NAV, then the ETF is said to be

trading "at a premium". If the price of the ETF is lower than the NAV, then the ETF is said to be trading "at a discount". If a financial trader detects a discrepancy between the ETF share price and the ETF NAV, then he/she can exploit that discrepancy to make a profit by way of arbitrage.

In one scenario, assume that the ETF share price, denoted by T_{ETF} , is higher than the ETF's NAV. In this case, a trader can execute the following steps:

- Assemble a basket by purchasing the shares of the financial instruments in the ETF's portfolio at a total cost approximately equal to the NAV;
- Create an ETF share by exchanging the assembled basket with the fund owner; and
- Sell the ETF on the exchange at a price approximately equal to T_{ETF} .

If the difference between T_{ETF} and the NAV is large enough to offset the transaction costs and other cash payments, then the trader can realize a profit by these actions. Moreover, this profit is realized by riskless arbitrage (assuming these steps could be performed instantaneously).

In another scenario, assume that the ETF share price, denoted by T_{ETF} , is lower than the ETF's NAV. In this case, a trader can purchase an ETF share, redeem it for the shares of the financial instruments in the portfolio, and sell those financial instrument shares on the exchange. Once again, if the difference between T_{ETF} and the NAV is large enough to offset the transaction costs and other cash payments, then the trader can realize a profit at virtually no risk.

The crucial element in profiting from such riskless arbitrages for ETFs is detecting the difference between the ETF's share price and the ETF's NAV. The first trader to detect such discrepancies will stand to benefit the most, and the latency in NAV computation becomes important - the computation of the NAV should be performed as quickly as possible to beat competitors in the race toward detecting profit opportunities.

Independently of the ETF scenarios discussed above, another utility of basket NAV calculation is that the NAV provides a measure of how the basket's portfolio is faring. A steadily increasing NAV indicates that investing in more such baskets may be profitable. On the other hand, a decreasing NAV indicates that it may be desirable to sell the basket portfolio in order to avoid loss. This strategy is similar to ones that would be employed when trading in the shares of an individual financial instrument with the basket providing a benefit by mitigating risk through diversification across a portfolio of financial instruments (e.g., if the basket as a whole has an increasing NAV, then investing in that basket will be desirable independently of how any individual financial instrument in the basket is performing).

In an effort to satisfy a need in the art for high volume low latency basket value computations, the inventors herein disclose a technique for computing basket values for a set of baskets based on the data within a high speed data stream as that data streams through a processor.

According to one aspect of a preferred embodiment of the present invention, the inventors disclose a technique for computing basket values wherein a coprocessor is used to perform the basket calculations. By offloading the computational burden of the basket calculations to a coprocessor, the main processor for a system can be freed to perform different tasks. The coprocessor preferably hardware accelerates the basket calculations using reconfigurable logic, such as Field Programmable Gate Arrays (FPGAs). In doing so, a preferred embodiment preferably harnesses the underlying hardware-accelerated technology disclosed in the following patents and patent applications: U.S. patent 6,711,558 entitled "Associated Database Scanning and Information Retrieval", U.S. Patent 7,139,743 entitled "Associative Database Scanning and Information Retrieval using FPGA Devices", U.S. Patent Application Publication 2006/0294059 entitled "Intelligent Data Storage and Processing Using FPGA Devices", U.S. Patent Application Publication 2007/0067108 entitled "Method and Apparatus for Performing Biosequence Similarity Searching", U.S. Patent Application Publication 2008/0086274 entitled "Method and

Apparatus for Protein Sequence Alignment Using FPGA Devices"

(published from U.S. Application Serial No. 11/836,947, filed August 10, 2007), U.S. Patent Application Publication 2007/0130140 entitled "Method and Device for High Performance Regular Expression Pattern Matching", U.S. Patent Application Publication 2007/0260602 entitled "Method and Apparatus for Approximate Pattern Matching", U.S. Patent Application Publication 2007/0174841 entitled "Firmware Socket Module for FPGA-Based Pipeline Processing", U.S. Patent Application Publication 2007/0237327 entitled "Method and System for High Throughput Blockwise Independent Encryption/Decryption", U.S. Patent Application Publication 2007/0294157 entitled "Method and System for High Speed Options Pricing", and pending U.S. Application Serial No. 11/765,306 entitled "High Speed Processing of Financial Information Using FPGA Devices", filed June 19, 2007, the entire disclosures of each of which are incorporated herein by reference.

According to another aspect of a preferred embodiment of the present invention, the inventors disclose that the different tasks involved in such basket calculations can be relegated to different modules within a pipeline. For example, one pipeline module can be configured to determine which baskets are impacted by a current data event within the stream. Another pipeline module can be configured to receive information about the impacted baskets and compute data indicative of a new basket value for each of those impacted baskets. By pipelining these modules together, the two modules can work on different messages and/or baskets at the same time in parallel with each other.

In a preferred embodiment, this pipeline comprises a basket association lookup module in communication with a basket value updating module. The basket association lookup module is configured to determine which baskets are impacted by each data message, while the basket value updating module is configured to compute at least one new value for each impacted basket based on the information within the data messages (for example, in an embodiment operating on financial market data, this information can be the financial instrument price information within the financial market data messages).

The basket value updating module preferably employs a delta calculation approach to the computation of the updated basket values, as explained in greater detail below. With such an approach, the number of arithmetic operations needed to compute each
5 basket value will remain the same regardless of how many elements such as financial instruments are members of a given basket. Furthermore, the basket value updating module may optionally be configured to compute a plurality of different types of basket values for the same basket simultaneously using parallel computation
10 logic.

The basket association lookup module preferably accesses a basket set table which stores at least a portion of the data needed by the basket value updating module for the basket value calculations. In a preferred embodiment pertaining to financial
15 information, this data can be indirectly indexed in the table by financial instrument such that, as a new message pertaining to a financial instrument is received, the appropriate data needed for the basket calculations can be retrieved. To provide the indirection, a second table is employed which indexes pointers into
20 the basket set table by financial instrument.

Furthermore, in an embodiment wherein the pipeline comprises a firmware pipeline deployed on a coprocessor such as a reconfigurable logic device, the design of these modules can be tailored to the hardware capabilities of the coprocessor, thereby providing a level
25 of synergy that streamlines the basket calculations so that the data streams can be processed at line speeds while a main processor is still free to perform other tasks.

The inventors also note that the pipeline can employ a price event trigger module downstream from the basket value updating
30 module. The price event trigger module can be configured to determine whether any of the updated basket values meet a client-specified trigger condition. Any updated basket values which meet the client-specified trigger condition can then be forwarded to the attention of the pertinent client (e.g., a particular trader, a
35 particular application, etc.). By doing so, clients can take

advantage of the low latency nature of the basket calculation pipeline to learn of trading opportunities which may be available.

Furthermore, the pipeline can employ an event generator module downstream from the price event trigger module, wherein the role of the event generator module is to generate a message event for delivery to the client in response to a trigger being set off by the price event trigger module. This message event preferably contains the updated basket value..

Further still, to limit the volume of financial market data messages processed by the basket association lookup module (and its downstream modules in the pipeline), the pipeline can employ a message qualifier filter module upstream from the basket association lookup module. Preferably, the message qualifier filter module is configured to drop messages from the financial market data message stream which are not pertinent to the basket calculations.

A pipeline such as the one described herein can be used in an embodiment of the invention to identify arbitrage conditions (such as the ETF arbitrage scenario discussed above) as they may arise in connection with financial instrument baskets and the current state of the markets. By detecting these arbitrage conditions quickly via the low latency basket calculation pipeline, a trader is enabled to make trades on one or more exchanges which take advantage of the detected arbitrage condition.

These and other features and advantages of the present invention will be apparent to those having ordinary skill in the art upon review of the following description and drawings.

Brief Description of the Drawings:

Figures 1(a) and (b) depict exemplary embodiments for a system on which hardware-accelerated basket calculations can be performed;

Figures 2(a) and (b) illustrate exemplary printed circuit boards for use as coprocessor 140;

Figure 3 illustrates an example of how a firmware pipeline can be deployed across multiple reconfigurable logic devices;

Figure 4(a) is a high level block diagram view of how a coprocessor can be used to perform basket calculations;

Figure 4(b) illustrates an exemplary process flow for computing an updated basket value using a delta calculation approach;

5 Figure 5 depicts an exemplary basket calculation engine pipeline in accordance with an embodiment of the invention;

Figure 6 depicts an exemplary basket association lookup module;

Figure 7 depicts an exemplary basket set pointer table;

Figure 8 depicts an exemplary basket set table;

10 Figure 9 depicts an exemplary basket value updating module;

Figures 10(a)-(c) depict exemplary embodiments for the NAV compute logic within the basket value updating module;

Figure 11 depicts an exemplary embodiment of the basket value updating module employing a plurality of NAV update engines;

15 Figure 12 depicts an exemplary basket calculation engine pipeline in accordance with another embodiment of the invention;

Figure 13 depicts an exemplary price event trigger module;

Figure 14 depicts an exemplary embodiment for the NAV-to-trigger comparison logic within the price event trigger module;

20 Figure 15 depicts an exemplary embodiment for the price event trigger module employing a plurality of client-specific price event trigger modules;

Figure 16 depicts an exemplary basket calculation engine pipeline in accordance with another embodiment of the invention;

25 Figure 17 depicts an exemplary event generator module;

Figure 18 depicts an exemplary basket calculation engine pipeline in accordance with yet another embodiment of the invention;

Figure 19 depicts an exemplary message qualifier filter module;

30 Figure 20 depicts an exemplary basket calculation engine pipeline in accordance with another embodiment of the invention; and

Figure 21 depicts an exemplary ticker plant architecture in which a basket calculation engine pipeline can be employed.

35 Detailed Description of the Preferred Embodiment:

Figure 1(a) depicts an exemplary embodiment for system 100 configured to perform high speed basket calculations. Preferably, system 100 employs a hardware-accelerated data processing capability through coprocessor 140 to perform basket calculations. Within
5 system 100, a coprocessor 140 is positioned to receive data that streams into the system 100 from a network 120 (via network interface 110). In a preferred embodiment, system 100 is employed to receive financial market data and perform basket calculations for financial instrument baskets. Network 120 thus preferably comprises
10 a network through which system 100 can access a source for financial data such as the exchanges themselves (e.g., NYSE, NASDAQ, etc.) or a third party provider (e.g., extranet providers such as Savvis or BT Radians). Such incoming data preferably comprises a series of financial market data messages, the messages representing events
15 such as trades and quotes relating to financial instruments. These messages can exist in any of a number of formats, as is known in the art.

The computer system defined by processor 112 and RAM 108 can be any commodity computer system as would be understood by those
20 having ordinary skill in the art. For example, the computer system may be an Intel Xeon system or an AMD Opteron system. Thus, processor 112, which serves as the central or main processor for system 100, preferably comprises a GPP.

In a preferred embodiment, the coprocessor 140 comprises a
25 reconfigurable logic device 102. Preferably, data streams into the reconfigurable logic device 102 by way of system bus 106, although other design architectures are possible (see Figure 2(b)). Preferably, the reconfigurable logic device 12 is a field programmable gate array (FPGA), although this need not be the case.
30 System bus 106 can also interconnect the reconfigurable logic device 102 with the processor 112 as well as RAM 108. In a preferred embodiment, system bus 106 may be a PCI-X bus or a PCI-Express bus, although this need not be the case.

The reconfigurable logic device 102 has firmware modules
35 deployed thereon that define its functionality. The firmware socket module 104 handles the data movement requirements (both command data

and target data) into and out of the reconfigurable logic device, thereby providing a consistent application interface to the firmware application module (FAM) chain 150 that is also deployed on the reconfigurable logic device. The FAMs 150i of the FAM chain 150 are
5 configured to perform specified data processing operations on any data that streams through the chain 150 from the firmware socket module 404. Preferred examples of FAMs that can be deployed on reconfigurable logic in accordance with a preferred embodiment of the present invention are described below.

10 The specific data processing operation that is performed by a FAM is controlled/parameterized by the command data that FAM receives from the firmware socket module 104. This command data can be FAM-specific, and upon receipt of the command, the FAM will arrange itself to carry out the data processing operation controlled
15 by the received command. For example, within a FAM that is configured to perform an exact match operation between data and a key, the FAM's exact match operation can be parameterized to define the key(s) that the exact match operation will be run against. In this way, a FAM that is configured to perform an exact match
20 operation can be readily re-arranged to perform a different exact match operation by simply loading new parameters for one or more different keys in that FAM. As another example pertaining to baskets, a command can be issued to the one or more FAMs that make up a basket calculation engine to add/delete one or more financial
25 instruments to/from the basket.

Once a FAM has been arranged to perform the data processing operation specified by a received command, that FAM is ready to carry out its specified data processing operation on the data stream that it receives from the firmware socket module. Thus, a FAM can
30 be arranged through an appropriate command to process a specified stream of data in a specified manner. Once the FAM has completed its data processing operation, another command can be sent to that FAM that will cause the FAM to re-arrange itself to alter the nature of the data processing operation performed thereby. Not only will
35 the FAM operate at hardware speeds (thereby providing a high throughput of data through the FAM), but the FAMs can also be

flexibly reprogrammed to change the parameters of their data processing operations.

The FAM chain 150 preferably comprises a plurality of firmware application modules (FAMs) 150a, 150b, ... that are arranged in a pipelined sequence. However, it should be noted that within the firmware pipeline, one or more parallel paths of FAMs 150i can be employed. For example, the firmware chain may comprise three FAMs arranged in a first pipelined path (e.g., FAMs 150a, 150b, 150c) and four FAMs arranged in a second pipelined path (e.g., FAMs 150d, 150e, 150f, and 150g), wherein the first and second pipelined paths are parallel with each other. Furthermore, the firmware pipeline can have one or more paths branch off from an existing pipeline path. A practitioner of the present invention can design an appropriate arrangement of FAMs for FAM chain 150 based on the processing needs of a given application.

A communication path 130 connects the firmware socket module 104 with the input of the first one of the pipelined FAMs 150a. The input of the first FAM 150a serves as the entry point into the FAM chain 150. A communication path 132 connects the output of the final one of the pipelined FAMs 150m with the firmware socket module 104. The output of the final FAM 150m serves as the exit point from the FAM chain 150. Both communication path 130 and communication path 132 are preferably multi-bit paths.

The nature of the software and hardware/software interfaces used by system 100, particularly in connection with data flow into and out of the firmware socket module are described in greater detail in the above-referenced and incorporated U.S. Patent Application Publication 2007/0174841.

Figure 1(b) depicts another exemplary embodiment for system 100. In the example of Figure 1(b), system 100 includes a data store 142 that is in communication with bus 106 via disk controller 114. Thus, the data that is streamed through the coprocessor 140 may also emanate from data store 142. Data store 142 can be any data storage device/system, but it is preferably some form of mass storage medium. For example, data store 142 can be a magnetic storage device such as an array of Seagate disks.

Figure 2(a) depicts a printed circuit board or card 200 that can be connected to the PCI-X or PCI-e bus 106 of a commodity computer system for use as a coprocessor 140 in system 100 for any of the embodiments of Figures 1(a)-(b). In the example of Figure 2(a), the printed circuit board includes an FPGA 102 (such as a Xilinx Virtex II FPGA) that is in communication with a memory device 202 and a PCI-X bus connector 204. A preferred memory device 202 comprises SRAM and DRAM memory. A preferred PCI-X or PCI-e bus connector 204 is a standard card edge connector.

Figure 2(b) depicts an alternate configuration for a printed circuit board/card 200. In the example of Figure 2(b), a bus 206 (such as a PCI-X or PCI-e bus), one or more disk controllers 208, and a disk connector 210 are also installed on the printed circuit board 200. Any commodity disk interface technology can be supported, as is understood in the art. In this configuration, the firmware socket 104 also serves as a PCI-X to PCI-X bridge to provide the processor 112 with normal access to any disk(s) connected via the private PCI-X bus 206. It should be noted that a network interface can be used in addition to or in place of the disk controller and disk connector shown in Figure 2(b).

It is worth noting that in either the configuration of Figure 2(a) or 2(b), the firmware socket 104 can make memory 202 accessible to the bus 106, which thereby makes memory 202 available for use by an OS kernel as the buffers for transfers to the FAMs from a data source with access to bus. It is also worth noting that while a single FPGA 102 is shown on the printed circuit boards of Figures 2(a) and (b), it should be understood that multiple FPGAs can be supported by either including more than one FPGA on the printed circuit board 200 or by installing more than one printed circuit board 200 in the system 100. Figure 3 depicts an example where numerous FAMs in a single pipeline are deployed across multiple FPGAs.

Figure 4(a) depicts at a high level a coprocessor 140 that receives an incoming stream of new financial instrument prices, T_j^{new} , and computes new basket values, NAV^{new} , in response to the received stream using a basket calculation engine 400. By

offloading the basket calculations to coprocessor 140, the system 100 can greatly decrease the latency of calculating new basket values in response to new financial instrument prices.

The basket calculation engine (BCE) 400 preferably derives its computation of NAV^{new} from formula (1). A direct implementation of formula (1) by BCE 400 would comprise M multiplications, M-1 additions, and 1 division. While a practitioner of the invention may choose to implement such a direct use of formula (1) within BCE 400, a preferred embodiment for BCE 400 reduces the number of arithmetic operations that need to be performed to realize the function of formula (1). Given that the basis upon which the NAV for a basket is to be updated is a new price for one of the basket's underlying financial instruments, it suffices to calculate the contribution to the basket's NAV of the difference between the new financial instrument price and the old financial instrument price. This calculated contribution can then be added to the old NAV value for the basket to find the updated NAV value. This process is referred to herein as a "delta calculation" for the NAV. This basis for this delta calculation approach is shown below, starting with formula (1).

$$\begin{aligned}
 NAV^{new} &= \frac{1}{d} \sum_{i=1}^M w_i T_i \\
 NAV^{new} &= \left(\frac{1}{d} \sum_{\substack{i=1 \\ i \neq j}}^M w_i T_i \right) + \left(\frac{1}{d} w_j T_j^{new} \right) \\
 NAV^{new} &= \left(\frac{1}{d} \sum_{\substack{i=1 \\ i \neq j}}^M w_i T_i \right) + \left(\frac{1}{d} w_j T_j^{new} \right) + \frac{1}{d} (w_j T_j^{old} - w_j T_j^{old}) \\
 NAV^{new} &= \frac{1}{d} \left(\left(\sum_{\substack{i=1 \\ i \neq j}}^M w_i T_i \right) + w_j T_j^{old} \right) + \frac{1}{d} (w_j T_j^{new} - w_j T_j^{old}) \\
 NAV^{new} &= \frac{1}{d} \left(\left(\sum_{\substack{i=1 \\ i \neq j}}^M w_i T_i \right) + w_j T_j^{old} \right) + \frac{w_j}{d} (T_j^{new} - T_j^{old})
 \end{aligned}$$

$$NAV^{new} = NAV^{old} + \Delta j \quad (2)$$

Thus, it can be seen that NAV^{new} can be computed as the sum of the old NAV price, NAV^{old} , and Δj , wherein Δj represents the delta contribution to the NAV of the new financial instrument price, and wherein Δj is computed as:

$$\Delta j = \frac{w_j}{d} (T_j^{new} - T_j^{old}) \quad (3)$$

Accordingly, it can be seen that the delta calculation approach can reduce the number of computations needed to calculate the new basket value from M multiplications, $M-1$ additions, and 1 division to only 1 subtraction, 1 multiplication, 1 division, and 1 addition. This reduction can lead to powerful improvements in computational efficiency because a single basket may contain an arbitrary number of financial instruments. It should be noted that, for some baskets, the value of M may be quite large. For example, a basket may be used to compute the Wilshire 5000 stock index, which is an index that includes approximately 6,300 securities. However, the presence of such a large number of securities within the Wilshire 5000 stock index would not add to the computational latency of a BCE 400 which employs the delta calculation approach because the NAV computation for such a basket will be based on formula (2) above.

Figure 4(b) depicts a high level process flow for the computation of the updated basket value by the BCE 400. At step 402, the BCE computes the delta contribution Δj of the new financial instrument price T_j^{new} to the basket value (see formula (3)). At step 404, the BCE computes the updated basket value NAV^{new} from the computed delta contribution (see formula (2)).

Figure 5 depicts a preferred pipeline 500 for realizing the BCE 400. Preferably, pipeline 500 is deployed as a firmware pipeline 150 on a reconfigurable logic device 102. The pipeline 500 preferably comprises a basket association lookup module 502 and a downstream basket value updating module 504. The basket association lookup module 502 receives a stream of financial market data messages, wherein each message represents a change in the bid, ask, and/or last price of a single financial instrument. Each message preferably comprises at least a symbol identifier, a global exchange

identifier, and one or more of a last price, a bid price, and an ask price for the financial instrument corresponding to the symbol identifier. The symbol identifier (or symbol ID) preferably comprises a binary tag that uniquely identifies a particular financial instrument. The global exchange identifier (GEID) preferably comprises a binary tag that uniquely identifies a particular exchange for which the message is relevant. Data tags within the messages preferably identify whether the price information within the message pertains to a bid/ask/last price for the financial instrument. Also, the message fields are preferably normalized as between the different message sources such prior to the time the messages reach the basket association lookup module 502.

The basket association lookup (BAL) module 502 is configured to determine, for each incoming message, a set of baskets which include the financial instrument that is the subject of that message. This basket set may comprise a plurality Q of baskets (although it should be noted that only one basket may potentially be present within the basket set). Thus, the volume heavy nature of tracking basket values for each financial market data message can be understood as each message causing a need for at least Q basket value calculations (a $1:Q$ expansion). In an exemplary embodiment, the maximum value for Q can be 1,024 baskets. However, other values could readily be used.

Figure 6 depicts an exemplary BAL module 502. System 100 preferably maintains a record for every known financial instrument, wherein each record preferably contains the list of baskets which contain that financial instrument, the relative weight of the financial instrument within each basket on the list, and the most recent bid, ask, and last sale price for the financial instrument. These records are preferably stored in a basket set table 608. System 100 also preferably maintains a basket set pointer table 604. Table 604 preferably comprises a set of pointers and other information that point to appropriate records in the basket set table 608. The use of the basket set pointer table 604 in combination with the basket set table 608 allows for more

flexibility in managing the content of the basket set table 608. In an embodiment wherein the coprocessor employs a memory 202 such as that shown in Figures 2(a) and (b), this memory 202 may comprise an SRAM memory device and an SDRAM memory device. Preferably, the basket set pointer table 604 can be stored in the SRAM memory device while the basket set table 608 can be stored in the SDRAM memory device. However, it should also be noted that, for an embodiment of the coprocessor wherein one or more FPGA chips are used, either or both of tables 604 and 608 can be stored in available on-chip memory should sufficient memory be available. Furthermore, tables 604 and 608 could be stored in other memory devices within or in communication with system 100.

In operation, the BAL module 502 performs a lookup in the basket set pointer table 604 using the symbol ID and GEID 600 for each message. Based on the symbol ID and GEID 600, a basket set pointer 606 is retrieved from table 604, and the BAL module 502 uses this basket set pointer 606 to perform a lookup in the basket set table 608. The lookup in the basket set table 608 preferably operates to identify the basket set 610 for the financial instrument identified by the symbol ID and identify the stored last/bid/ask prices 612 for that financial instrument. Each basket set comprises identifiers for one or more baskets of which that financial instrument is a member.

A subtractor 616 preferably operates to subtract the retrieved last/bid/ask prices 612 from the new last/bid/ask prices 602 contained in the current message to thereby compute the changes in the last/bid/ask prices 614, as shown in Figure 6. It should be noted that each message may comprise one or more price fields for the subject financial instrument. For any price fields not contained within the message, the resultant price change is preferably treated as zero. Thus, if a message does not contain the last sale price for the financial instrument, then the Δ Last price will be zero.

Figure 7 depicts an exemplary embodiment for the basket set pointer table 604 in greater detail. Table 604 comprises a plurality of records 700, wherein each record comprises a bit string

corresponding to a set of flags 702, a bit string corresponding to a header block pointer 704, and a plurality of bit strings corresponding to different GEIDs 706. Each record 700 is keyed by a symbol ID such that the BAL module 502 is able to retrieve the record 700 from table 604 which corresponds to the symbol ID of the current message. The flags 702 denote whether the record 700 is valid (i.e., whether the subject financial instrument is contained within at least one basket). The header block pointer 704 serves as a pointer to a basket association record in table 608 for the financial instrument corresponding to the symbol ID.

Also, a given financial instrument may be fungible on multiple financial exchanges. The state of a financial instrument on a given exchange (e.g., the NYSE) is referred to as the regional view for that financial instrument. The aggregate state of that financial instrument across all exchanges is referred to as the composite view for that financial instrument. The composite value for the last price is preferably the most recent sales price for the financial instrument on any exchange. The composite value for the bid price is preferably the best of the bid prices for the financial instrument across all of exchanges on which that financial instrument is traded. The composite view for the ask price is preferably the best of the ask prices for the financial instrument across all of exchanges on which that financial instrument is traded. Thus, the bid price and ask price in the composite view are commonly referred to as the Best Bid and Offer (BBO) prices. Tables 604 and 608 allow for baskets to be defined with regional and/or composite views of financial instruments. For example, basket A may contain the composite view of the 10 technology stocks with the largest market capitalization. Basket B may contain the NYSE regional view of the 10 technology stocks with the largest market capitalization. Basket C may contain the composite view for 10 technology stocks and the NYSE regional view for 10 industrial stocks.

To accommodate such flexible basket definitions, table 604 preferably employs a plurality of GEID fields 706 within each record 700. Each GEID field 706 serves as a further index into table 608

to retrieve the appropriate composite basket association record and/or regional basket association record for the pertinent financial instrument. Thus, BAL module 502 uses the GEID of the current message to identify a matching GEID field 706 in record 700. Field 706₀, which corresponds to GEID₀, is preferably used to identify any basket association records in table 608 which correspond to the composite view for the financial instrument. The other fields 706 are used to identify any basket association records in table 608 which correspond to the region-specific view for the financial instrument. For example, GEID₁ may correspond to the NYSE view for the financial instrument while GEID_m may correspond to the NASDAQ view for the financial instrument. The matching GEID field 706 is then used as an additional index into table 608 to retrieve a set of basket association records for the financial instrument that are applicable to both composite views and regional views of the financial instrument.

Preferably, BAL module 502 will always retrieve the composite GEID₀ field applicable to a given financial instrument. Furthermore, for a message with regional price information about a financial instrument, the BAL module 502 also operates to match the GEID field in the message with a GEID field 706 in record 700 for that financial instrument. Thus, the basket set pointer 606 used by the BAL module 502 preferably comprises the header block pointer 704, the composite GEID index 706₀, and the regional GEID index 706_i for the subject message.

Figure 8 depicts an exemplary embodiment for the basket set table 608 in greater detail. Table 608 comprises a plurality of basket association records, wherein each record contains basket information for a different financial instrument. Each record preferably comprises a plurality of header blocks 800 and possibly one or more extension blocks 822. Each header block 800, which preferably has a fixed size, corresponds to the subject financial instrument and a particular GEID. Furthermore, each header block 800 preferably comprises a bit string corresponding to the GEID field 802 for that block, a bit string corresponding to an extension pointer 804, a bit string corresponding to a count field 806, a set

of reserved bits 808, a bit string that corresponds to the most recently stored bid price 810 for the subject financial instrument, a bit string that corresponds to the most recently stored ask price 812 for the subject financial instrument, a bit string that
5 corresponds to the most recently stored last price for the subject financial instrument 814, and a plurality of bit strings corresponding to a plurality of sets of basket information 816.

Preferably, each block 800 preferably comprises either composite basket association information or regional basket
10 association information for the subject financial instrument. Also, the blocks 800 are preferably located in memory such that the composite and regional blocks for a given financial instrument are stored in contiguous memory addresses. In this way, the header block pointer 704 can be used to locate the financial instrument's
15 composite block 800, while the GEID index 706 can be used to locate the applicable regional block 800 for the financial instrument by serving as an increment to the address found in the header block pointer 704.

Each basket information field 816 in table 608 preferably
20 comprises a basket identifier and weight pair (Basket ID, Weight). The Basket ID serves to identify a particular basket while the weight serves to identify the weight to be used when computing the delta contribution for subject financial instrument to the identified basket.

Given that a single financial instrument may be present in a number of different baskets, it is possible that the number of (Basket ID, Weight) pairs needed for a given financial instrument will not fit within a single block 800. To accommodate such
25 overflow situations, table 608 also preferably comprises a plurality of extension block records 822. Each extension block record 822 preferably has a fixed size and comprises a bit string corresponding to an extension pointer 818, a bit string corresponding to a count field 820, any one or more (Basket ID, Weight) pairs 816 as needed for the financial instrument. If the number of (Basket ID, Weight)
30 pairs 816 for the financial instrument are unable to fit within a single extension block 822, then the extension pointer 818 will
35

serve to identify the address of an additional extension block 822 in which the additional (Basket ID, Weight) pairs 816 are found. It should thus be noted that a plurality of extension blocks 822 may be needed to encompass all of a financial instrument's relevant (Basket ID, Weight) pairs 816. The count field 820 identifies how many (Basket ID, Weight) pairs 816 are present within the subject extension block 822.

The count field 806 within a block 800 serves to identify how many (Basket ID, Weight) pairs 816 are present within that block 800.

Fields 810, 812, and 814 contain the most recently stored bid/ask/last prices for the subject financial instrument on the pertinent exchange. As shown by Figure 6, this price information is used for computing the price change values 614 (or delta prices) that are relevant to the delta contribution calculation.

Thus, using the pointer 704 and GEID index reference 706 retrieved from table 604, the BAL module 502 is configured to access a block 800 in table 608 corresponding to the composite information relevant to the subject message (e.g., the block 800 shown in Figure 8 labeled with "GEID₀") and a block 800 in table 608 corresponding to the regional information relevant to the subject message (e.g., the record 800 shown in Figure 8 labeled with "GEID₁"). To find the appropriate regional block, the BAL module uses the matching GEID field 706 to count down from the pertinent composite block to the appropriate regional block. Thus, if the pertinent regional block for a particular financial instrument is the NYSE block, wherein the NYSE has a GEID value of that matches the second GEID field 706, GEID₂, in record 700, then the BAL module will find the appropriate NYSE block 800 in table 608 two blocks down from the composite block.

In the example of Figure 8, the regional information has an overflow condition, and the extension pointer 804 points to an extension block 822, as shown. Based on these lookups, the BAL module 502 retrieves the bid/ask/last prices and (Basket ID, Weight) pairs for the composite view of the financial instrument as well as the bid/ask/last prices and (Basket ID, Weight) pairs for the

appropriate regional view of the financial instrument. It should be noted that the retrieved bid/ask/last prices 612 for the composite view will need to be associated with the retrieved list of (Basket ID, Weight) pairs 610 for the composite view and the retrieved
5 bid/ask/last prices 612 for the regional view will need to be associated with the retrieved list of (Basket ID, Weight) pairs 610 for the regional view to ensure that downstream computations are based on the appropriate values. To maintain this association, delta events as described below are preferably employed.

10 BAL module 502 is also preferably configured to update the records in table 608 with the new prices 602 in each message. This is shown by arrow 618 in Figure 6, which represents a write operation into the appropriate fields of table 608. However, it should be noted that if the message does not include a particular
15 price field such as the last price field, then the BAL module 502 preferably maintains the last price field 814 in the pertinent block 800 in its existing state.

It should be noted that the array of GEID fields 706 for each record 700 in table 604 may be of fixed size for simplicity.

20 Depending upon how much size a practitioner allocates to the GEID fields 706 in each record 700, it may be the case that a financial instrument trades on a sufficiently large number of exchanges that there are not a sufficient number of fields 706 for storing the financial instrument's relevant GEIDs. Thus, if this situation
25 arises and the GEID 600 of the message does not find a match to any GEID fields 706 in the pertinent record 700 of table 604, then the BAL module 502, to retrieve an appropriate regional block from table 608, preferably indexes to the last header block 800 in the record for that financial instrument and conducts a linear search through
30 the header blocks' GEID fields 802 to find a match to the GEID 600 associated with the message.

It should also be noted that if the GEID 600 is not found in any of the GEID fields 802 for the financial instrument record in table 608, this means that the regional view of that financial
35 instrument corresponding to GEID 600 is not included in any baskets.

It should also be noted that baskets may be dynamically changed throughout the day by adding and/or removing (Basket ID, Weight) pairs from the basket association records in table 608 for financial instruments.

5 To add a financial instrument to a basket, the system adds the appropriate (Basket ID, Weight) pair to that financial instrument's record in table 608. A control process, preferably performed in software executing on a GPP such as processor 112, preferably maintains a copy of tables 604 and 608 and computes where entries
10 need to be added to the table 608 (and possibly table 604 if the GEID for that financial instrument is not present in the GEID fields 706 of record 700 for that financial instrument) to reflect the addition of the financial instrument to the basket. These determinations can be made using the symbol ID and GEID for the
15 financial instrument to be added. The control process then preferably sends memory write commands to the pipeline for consumption by the BAL module 502 that are effective to write update tables 604 and 608 to reflect the addition of the subject financial instrument to a basket. The control process may also be configured
20 to immediately send a synthetic event to the pipeline for consumption by the BAL module 502 wherein this event contains the current prices for the relevant financial instrument. This action will cause delta updates to the NAV values for the subject basket wherein these delta updates are relative to a zero (so the entirety
25 of the price contribution is determined). However, it should also be noted that the control process may be configured to simply wait for subsequent market events to arrive at the pipeline and allow the delta updates to be applied at that time.

Removing a financial instrument from a basket generally
30 involves removing the (basket ID, weight) pair for the subject basket from the record in table 608 for the subject financial instrument. Furthermore, to appropriately update the NAV values to reflect the deletion of the financial instrument from the basket, the control process preferably generates a synthetic event for
35 delivery to the pipeline wherein the price information for the subject financial instrument is zero. This causes the delta price

values 614 for the financial instrument to be the negative value of the price information for that financial instrument stored in table 608, thus removing the entirety of the price contribution for that financial instrument to the subject basket.

5 To add an entire basket to the system, it follows that the control process can issue appropriate memory write commands to the pipeline that adds (Basket ID, Weight) pairs to table 608 for all of the new basket's constituent financial instruments. The control process can also generate synthetic events for delivery to the
10 pipeline which reflects the current price information for these constituent financial instruments, as explained above. Moreover, as can be understood in connection with Figures 9 and 13 below, the control process can initiate the addition of appropriate entries to the divisor table, NAV tables, client NAV tables, and client trigger
15 threshold tables to reflect the addition of the new basket to the system.

 To delete an entire basket from the system, it also follows that the control process can be configured to initiate a removal of the (Basket ID, Weight) pairs from table 608 for all of the subject
20 basket's constituent financial instruments. Similarly, the records in the tables shown in Figures 9 and 13 can also be updated to remove those records associated with the Basket ID of the removed basket.

 The output from the BAL module 502 will be a stream of delta
25 events. Each delta event preferably comprises a retrieved (Basket ID, Weight) pair 610 and at least one computed price change 614 for a financial instrument corresponding to that retrieved (Basket ID, Weight) pair 610. In an embodiment of the invention, each delta event includes all of the price deltas, even if one or more of the
30 price deltas (e.g., the Δ Bid price) is zero. However, this need not be the case, as noted below.

 It should also be understood that, for each message event that is received at the input to the BAL module 502, a plurality of delta events may be produced at the output of the BAL module 502 due to
35 the fact that the financial instrument which is the subject of the received message may be a member of a plurality of baskets. Partly

because of this expansion, the low latency nature of pipeline 500 is particularly advantageous in allowing the task of updating basket values to keep up with the inflow of events from the different exchanges.

Returning to Figure 5, the basket value updating module 504 is configured to compute the new basket value for each delta event produced by the BAL module 502. Figure 8 depicts an exemplary basket value updating (BVU) module 504.

A delta event buffer 900 buffers the delta events produced by the BAL module 502. For each delta event read out of buffer 900, the BVU module 504 performs a lookup in a divisor table 902 to determine the appropriate value for the divisor d from formula (3) for the basket corresponding to the Basket ID within the current delta event. Thus, it can be seen that table 902 preferably indexes each divisor value by the Basket ID for the divisor's corresponding basket. The delta event buffer 900 and the divisor table 902 are preferably located within available on-chip memory for the reconfigurable logic device 102. However, it should be noted that buffer 900 and table 902 could alternatively be stored on any memory resource within or accessible to coprocessor 140.

A NAV update engine 950 then receives as an input the delta event information (the (Basket ID, Weight) pair 610 and the delta bid/ask/last prices 614) plus the divisor value d for that delta event to compute at least one updated basket value NAV^{new} .

Preferably, the NAV update engine 950 is configured to compute a plurality of different types of basket NAV's for each basket. For example, the NAV update engine 950 can be configured to compute, in parallel, new NAVs based on bid prices, ask prices, last prices, bid-tick prices, and ask+tick prices ($Bid\ NAV^{new}$, $Ask\ NAV^{new}$, $Last\ NAV^{new}$, $Bid-Tick\ NAV^{new}$, and $Ask+Tick\ NAV^{new}$ respectively), as shown in Figure 9. In these examples, the tick value is defined as the minimum increment used by the pertinent exchange to measure a change in financial instrument price.

A demultiplexer 904 operates to route portions of the delta event information and the divisor value to the appropriate NAV compute logic 916. The NAV update engine preferably employs a

plurality of parallel computing paths to compute each type of basket NAV.

The computing path for computing $Bid\ NAV^{new}$ preferably operates on the Basket ID, Weight, Divisor, and ΔBid price. The BVU module 504 performs a lookup in a Bid NAV table 906 based on the Basket ID to retrieve a stored Bid NAV value ($Bid\ NAV^{old}$) for the subject basket. This $Bid\ NAV^{old}$ value serves as the NAV^{old} value in formula (2) during computation of $Bid\ NAV^{new}$. NAV compute logic 916 then (i) computes the delta contribution Δj for the financial instrument according to formula (3) using the divisor value as the d term in formula (3), the Weight value as the w_j term in formula (3), and the ΔBid value as the $(T_j^{new} - T_j^{old})$ term in formula (3), and (ii) thereafter computes $Bid\ NAV^{new}$ according to formula (2) based on the computed Δj value and the $Bid\ NAV^{old}$ value.

The computing path for computing $Ask\ NAV^{new}$ preferably operates in the same manner as the computing path for $Bid\ NAV^{new}$ albeit performing a lookup in an Ask NAV table 908 based on the Basket ID and using the ΔAsk price rather than the ΔBid price.

The computing path for computing $Last\ NAV^{new}$ also preferably operates in the same manner as the computing path for $Bid\ NAV^{new}$ albeit performing a lookup in an Last NAV table 910 based on the Basket ID and using the $\Delta Last$ price rather than the ΔBid price.

Likewise, the computing paths for computing the $Bid-Tick\ NAV^{new}$ and $Ask+Tick\ NAV^{new}$ value preferably operate in the same manner as the computing path for $Bid\ NAV^{new}$ albeit performing lookups in, respectively, a Bid-Tick NAV table 912 and an Ask+Tick NAV table 914 based on the Basket ID and using, respectively, the ΔBid price and the ΔAsk price.

Table 912 can store previously computed NAVs for each basket that are based on a price for the financial instrument that is the previous bid price for the financial instrument minus a tick value. Table 914 can store previously computed NAVs for each basket that are based on a price for the financial instrument that is the previous ask price for the financial instrument plus a tick value.

Preferably, tables 906, 908, 910, 912, and 914 are stored in available on-chip memory for the reconfigurable logic device 102. However, it should be noted that these tables could alternatively be stored on any memory resource within or accessible to coprocessor 140.

Thus, it can be seen that the NAV update engine 950 is preferably configured to output, each clock cycle, an updated basket event, wherein each updated basket event comprises a plurality of different updated NAVs for a particular basket together with a Basket ID for that basket.

The BVU module is also preferably configured to update tables 906, 908, 910, 912, and 914 with the newly computed NAV values. Thus, the newly computed $Bid\ NAV^{new}$ for a given Basket ID can be fed back into an entry for that Basket ID in table 906 so that the next delta event pertaining to the same Basket ID will retrieve the most current NAV^{old} value. The memory for tables 906, 908, 910, 912, and 914 is preferably dual-ported, thus allowing for two read/write operations to occur on each clock cycle. If updating logic for the BVU module is pipelined, it can be expected that contention for the same Basket ID entry in a given table during a read and write operation on the same clock cycle will be rare and can be identified and avoided using techniques known in the art.

Also, a control process can issue a memory write command to the pipeline for consumption by the BVU module to update an entry in the divisor table 902 in the event of a change in value for a basket's divisor. Such a control process is preferably performed in software executing on a GPP such as processor 112.

Figure 10(a) depicts an exemplary embodiment for the NAV compute logic 916. A multiplier 1000 operates to compute the $w_j(T_j^{new} - T_j^{old})$ portion of formula (3) using the Weight value and delta price information from each delta event. A divider 1002 operates to divide the $w_j(T_j^{new} - T_j^{old})$ value by d to thereby compute Δj according to formula (3). Thereafter, an adder 1004 is used to add Δj to the NAV^{old} value, thereby computing NAV^{new} according to formula (2).

Figure 10(b) depicts another exemplary embodiment for the NAV compute logic 916. In the embodiment of Figure 10(b), the division

operation is performed last, thereby improving the precision of the NAV computation relative to the embodiment of Figure 10(a). To re-order the arithmetic operations such that the division operation inherent in formula (2) by way of formula (3) is performed last, the embodiment of Figure 10(b) preferably adds a multiplication operation using multiplier 1006. This multiplication operation can be performed in parallel with the multiplication operation performed by multiplier 1000. Multiplier 1006 operates to multiply the NAV^{old} value by d , thereby resulting in the value S^{old} . In this embodiment, the delta contribution of the financial instrument can be expressed as $w_j(T_j^{new} - T_j^{old})$, as shown at 1012 in Figure 10(b). Adder 1004 then computes the sum of S^{old} and $w_j(T_j^{new} - T_j^{old})$. Given that S^{old} equals $dNAV^{old}$, it can readily be seen that the act of dividing $S^{old} + w_j(T_j^{new} - T_j^{old})$ by d will result in NAV^{new} in accordance with formulas (2) and (3).

Figure 10(c) depicts yet another exemplary embodiment for the NAV compute logic 916. In the embodiment of Figure 10(c), the precision improvement of the Figure 10(b) embodiment is preserved while also eliminating the need for multiplier 1006. To do so, in a BVU module which employs the NAV compute logic 916 of Figure 10(c), the NAV tables 906, 908, 910, 912, and 914 preferably store S^{old} values rather than NAV^{old} values, thereby eliminating the need for multiplier 1006. Furthermore, to update tables 906, 908, 910, 912, and 914, the value of S^{new} (which is the same as $S^{old} + w_j(T_j^{new} - T_j^{old})$) is fed back into the relevant tables via communication link 1010 such that S^{new} serves as S^{old} when the next delta event is processed.

It should be noted that the BVU module 504 can employ a plurality of NAV update engines 950 in parallel to thereby simultaneously compute a plurality of basket NAV types for a plurality of baskets. An example of this is shown in Figure 11, wherein the BVU module 504 comprises two parallel NAV update engines 950. In such an embodiment, the BVU module 504 reads two delta events out of buffer 900 each clock cycle. After lookups are performed in the divisor table 902 to identify the appropriate divisor value for each delta event, routing logic 1100 routes the

delta event information and its associated divisor value to an appropriate NAV updated engine 950. Such routing logic 1100 can be configured to assign different segments of the Basket ID range to each set of parallel engines 950.

5 If desired, it should be noted that a single set of NAV tables 906, 908, 910, 912, and 914 can be shared by a plurality of NAV update engines 950. If the number of NAV update engines exceeds the number of available ports for reading from the NAV tables, then the BVU module 504 can be configured to interleave the accesses to these
10 tables from the different NAV update engines. These tables could also be replicated if desired to avoid such interleaving. The same can be said with respect to divisor table 902 (while Figure 11 depicts an embodiment wherein two divisor tables are utilized, it should be noted that a single divisor table 902 may be shared by the
15 different paths in Figure 11).

Figure 12 depicts an embodiment wherein pipeline 500 includes a price event trigger module 1200 in communication with the output from the BVU module 504. The price event trigger (PET) module 1200 is configured to determine which updated basket values produced by
20 the BVU module 504 are to be reported to an interested client. These determinations can be made on the basis of any of a number of criteria. The client may be a particular trader who has requested that he/she be notified when the value of a basket changes by a certain amount, or the client may be another application within a
25 trading system. The client may also be another module deployed on the coprocessor 140 (such as another FAM 150 if pipeline 500 is a FAM pipeline).

Figure 13 depicts an exemplary embodiment for a PET module 1200. Preferably, PET module 1200 is configured to operate in
30 parallel on each basket value type for a basket generated by the BVU module 504. Also, to determine which of the updated basket values should be reported to interested clients, the PET module 1200 preferably compares each updated basket value with a client-defined trigger threshold.

35 PET module 1200 preferably accesses a plurality of tables corresponding to different NAV types, wherein these tables store

client reference values for each basket. These reference values are used by the PET module 1200 to judge whether new basket values should be reported to interested clients. In an exemplary embodiment, the reference values are NAV values for the subject

5 baskets which were previously reported to the client. Thus, a client NAV table preferably exists for each basket value type generated by the BVU module 504. Thus, reference value table 1302 stores a plurality of client Bid NAVs for the different baskets, wherein each client Bid NAV is indexed by a basket identifier.

10 Reference value table 1304 stores a plurality of client Ask NAVs for the different baskets (each client Ask NAV being indexed by a basket identifier). Reference value table 1306 stores a plurality of client Last NAVs for the different baskets (each client Last NAV being indexed by a basket identifier). Reference value table 1308

15 stores a plurality of client Bid-Tick NAVs for the different baskets (each client Bid-Tick NAV being indexed by a basket identifier).

Reference value table 1310 stores a plurality of client Ask+Tick NAVs for the different baskets (each client Ask+Tick NAV being indexed by a basket identifier). Each of these tables preferably

20 stores the most recent of their respective NAV types to be reported to the client. Thus, in the event of one of the triggers being set off by a new basket value, the PET module is preferably configured to update the NAV values in the client tables with the new basket values. Such updating logic can be configured to update the tables

25 only for those NAV values which set off the trigger or can be configured to update all of the tables when any NAV value sets off a trigger.

Also, a client NAV trigger threshold table preferably exists for each basket value type generated by the BVU module 504. Thus,

30 table 1314 stores a plurality of client Bid NAV trigger threshold values for the different baskets, wherein each client Bid NAV trigger threshold value is indexed by a basket identifier. Table 1316 stores a plurality of client Ask NAV trigger threshold values for the different baskets (each client Ask NAV trigger threshold

35 value being indexed by a basket identifier). Table 1318 stores a plurality of client Last NAV trigger threshold values for the

different baskets (each client Last NAV trigger threshold value being indexed by a basket identifier). Table 1320 stores a plurality of client Bid-Tick NAV trigger threshold values for the different baskets (each client Bid-Tick NAV trigger threshold value being indexed by a basket identifier). Table 1322 stores a plurality of client Ask+Tick NAV trigger threshold values for the different baskets (each client Ask+Tick NAV trigger threshold value being indexed by a basket identifier).

Preferably, tables 1302, 1304, 1306, 1308, 1310, 1314, 1316, 1318, 1320, and 1322 are stored in available on-chip memory for the reconfigurable logic device 102. However, it should be noted that these tables could alternatively be stored on any memory resource within or accessible to coprocessor 140.

Also, it should be noted that the client NAV tables and the client NAV trigger threshold tables can be optionally consolidated if desired by a practitioner of this embodiment of the invention. In doing so, the records in the client Bid NAV table 1302 would be merged with the records in the client Bid NAV trigger threshold table 1314 such that each record comprises the client Bid NAV and the client Bid NAV trigger threshold value. Similarly, table 1304 could be merged with table 1316, table 1306 with table 1318, table 1308 with table 1320, and table 1310 with table 1322.

PET module 1200 preferably employs a plurality of parallel computing paths to determine whether any of the NAV types within each basket event set off a client trigger.

A first computing path is configured to perform a lookup in the client Bid NAV table 1302 and a lookup in the client Bid NAV trigger threshold table 1314 based on the current updated basket event's Basket ID to thereby retrieve a client Bid NAV and a client Bid NAV trigger threshold value. The client Bid NAV serves as a trigger against which the Bid NAV within the updated basket event is compared. The client Bid NAV trigger threshold value is then used as the criteria to judge whether the trigger has been set off. NAV-to-trigger comparison logic 1312 then operates perform this comparison and determine whether the trigger has been set off,

wherein a Bid NAV trigger flag indicates the result of this determination.

Additional computing paths perform these operations for the updated basket event's Ask NAV, Last NAV, Bid-Tick NAV, and Ask+Tick NAV to produce Ask NAV, Last NAV, Bid-Tick NAV, and Ask+Tick NAV trigger flags, respectively, as shown in Figure 13.

Trigger detection logic 1324 receives the different NAV trigger flags produced by the computing paths. This logic 1324 is configured to determine whether any of the trigger flags for a given updated basket event are high. Preferably, the trigger detection logic 1324 is configured pass an updated basket event as an output if any of the flags for that updated basket event are high. As such, it can be seen that the PET module 1200 serves as a filter which passes updated basket events that are deemed noteworthy according to client-defined criteria.

Figure 14 illustrates an exemplary embodiment for the comparison logic 1312. A subtractor 1400 operates to subtract the retrieved client NAV from the updated basket event's NAV (NAV^{new}). A comparator 1404 then compares the difference between NAV^{new} and the client NAV with the retrieved client trigger threshold to determine whether the NAV trigger flag should be set. If the difference exceeds the threshold value, then comparator preferably sets the NAV trigger flag to high.

Thus, a client may want to be notified whenever the Bid NAV of Basket 145 changes by \$0.05, whenever the Ask NAV for Basket 145 changes by \$0.03, or whenever the Last NAV for Basket 145 changes by \$0.02. In such an instance, the client Bid NAV trigger threshold value in table 1314 for Basket 145 can be set equal to \$0.05, the client Ask NAV trigger threshold value in table 1316 for Basket 145 can be set equal to \$0.03, and the client Last NAV trigger threshold value in table 1318 for Basket 145 can be set equal to \$0.02. If the new bid NAV, Ask NAV, and Last NAV values within the updated basket event for Basket 145 are \$20.50, \$20.40, and \$20.20 respectively, while the stored values for the client NAVs of Basket 145 are \$20, \$20.42, and \$20.19 respectively, then the comparison

logic will operate to set the Bid NAV trigger flag high, the Ask NAV trigger flag low, and the Last NAV trigger flag high.

However, it should be noted that the comparison logic 1312 could be configured to determine whether a client trigger has been set off in any of a number of ways. For example, the client threshold values can be expressed in terms of percentages if desired (with corresponding adjustments in the comparison logic to compute the percentage difference between the new NAV and the client NAV). Also, the trigger condition could be based on a tracking instrument that is different than the subject NAV. For example, the trigger condition for a given basket can be defined in relation to a stored price for the DJIA. The client NAV tables would then store a DJIA price for that basket, and that basket's NAV prices would be compared against the retrieved DJIA price to determine whether the trigger threshold has been reached.

It should be noted that the PET module 1200 can be configured to detect arbitrage conditions by appropriately populating the client NAV tables and the client trigger threshold tables. For example, with ETFs, a client NAV table can be populated with the current share price of the ETF itself (T_{ETF}). Thus, the client bid NAV table could be populated with the current bid price for a share of the ETF, the client ask NAV table could be populated with the current ask price for a share of the ETF, the client last NAV table could be populated with the current last price for a share of the ETF, etc. By populating the client NAV tables in this manner, the PET module can detect when there is a discrepancy between the share price of the ETF and the net asset value of the ETF's constituent financial instruments. Such a discrepancy can be taken advantage of by a trader, as explained above.

Thus, a client application that is informed by the pipeline of a discrepancy between T_{ETF} and the ETF's NAV can place one or more trades on one or more financial markets to realize a profit from the detected arbitrage condition. For example, if the ETF's NAV is less than the ETF's share price, a client application can execute the following steps:

- Assemble a basket by purchasing the shares of the financial instruments in the ETF's portfolio at a total cost approximately equal to the NAV;
- Create an ETF share by exchanging the assembled basket with the fund owner; and
- Sell the ETF on the exchange at a price approximately equal to T_{ETF} .

Furthermore, if the ETF's NAV is greater than the ETF's share price, a client application can execute the above steps in reverse. That is, a trader can buy one or more shares of the subject ETF from the market, exchange the ETF for shares of its constituent financial instruments, and then sell those shares of the constituent financial instruments on the market.

It should also be noted that the trigger thresholds in the client trigger threshold tables can be populated with values that take into account any expected transactional costs involved in the process of making such trades to take advantage of the price discrepancy, as explained below. Thus, if the sales price for the assembled ETF exceeds the aggregate prices of the financial instrument shares purchased to assemble the basket and the related transactional costs in doing so (or the sales prices for the constituent financial instruments exceeds the purchase price of the ETF and the related transactional costs), a trader who uses the client application can yield a profit by way of arbitrage. For such actions to work, the inventors note that time is of the essence, and it is believed that the low latency nature of the disclosed pipeline greatly increases a trader's ability to recognize and profit from such arbitrage conditions as they arise in a constantly fluctuating market.

Moreover, it should be noted that the PET module 1200 can filter updated basket events for a plurality of different clients. In this manner, each subscribing client can define its own set of reference values and trigger thresholds. To accomplish such a feature, each updated basket event can be compared in parallel with a plurality of criteria for different clients via a plurality of client-specific PET modules 1500, as shown in Figure 15. It should

also be noted that the client-specific PET modules 1500 (each of which would be a client-specific replicant of PET module 1200 of Figure 13) can be arranged in series (wherein each client-specific PET module would optionally add a client-specific flag to each
5 updated basket event to thereby indicate whether that client's triggering criteria were met). Furthermore, the client-specific PET modules 1500 could also be arranged in a hybrid parallel/series arrangement if desired.

Figure 16 depicts an embodiment wherein pipeline 500 includes
10 an event generator module 1600 in communication with the output from the PET module 1200. The event generator module 1600 is configured to process each incoming triggered basket event to construct a normalized output message event which contains the computed new NAVs within the triggered basket event.

Figure 17 depicts an exemplary embodiment for the event
15 generator module 1600. Preferably, the event generator module 1600 is configured to insert a bit string into each output event 1704 which defines an interest list for that event, wherein the interest list identifies each client that is to be notified of the event.
20 Preferably, this interest list takes the form of a bit vector, wherein each bit position corresponds to a different client. For any bit positions that are high, this means that the client corresponding to that bit position should be notified.

An interest list table 1700 preferably stores such a bit
25 vector for each basket. As the event generator module 1600 receives a triggered basket event from the PET module 1200, the event generator module 1600 performs a lookup in the interest list table 1700 based on the Basket ID within the triggered basket event. As a result of this lookup, the interest list vector for that Basket ID
30 is retrieved. A formatter 1702 thereafter builds an output message event 1704 from the retrieved interest list vector 1706 and the information within the received triggered basket event, as shown in Figure 17. Formatter 1702 is preferably configured to add a header 1708 and trailer 1710 to the output event 1704 such that the output
35 event 1704 can be properly interpreted by a client recipient (or recipients) of the output event 1704. The format for such output

events can comply with any of a number of messaging protocols, both well-known (e.g., FAST and FIX) and proprietary. As indicated, the value of the bit positions in the interest list vector 1706 of the output event 1704 will define which clients are to be recipients of the event 1704.

The interest list table 1700 is preferably located within available on-chip memory for the reconfigurable logic device 102. However, it should be noted that table 1700 could alternatively be stored on any memory resource within or accessible to coprocessor 140. Also, it should be noted that formatter 1702 may optionally be configured to access one or more data tables (not shown; either on-chip or off-chip) to define the appropriate fields for the output event 1704.

It should also be noted that the event generator module 1600 may be configured to store statistics such as the number of triggered basket events it receives, the number of output events it produces, and the average inter-event time for each triggered basket event.

Figure 18 depicts an embodiment wherein pipeline 500 includes a message qualifier filter module 1800 in communication with the input to the BAL module 502. The message qualifier filter module 1800 is configured to process incoming message events and filter out any message events that are not of interest to the BCE. Messages of interest will generally include messages which may affect NAV prices. Examples of messages that a practitioner of this embodiment of the invention may want to filter are messages that do not contain a bid, ask, or last price for a financial instrument. For example, a practitioner may desire to filter out messages which merely correct the previous day's closing prices and messages relating to trades that do not contribute to the daily volume.

Figure 19 depicts an exemplary message qualifier filter module 1800 that can be employed in the pipeline 500 of Figure 18. A table 1900 stores a plurality of qualifiers. Module 1800 is preferably configured to compare one or more fields of each incoming message (such as the quote and trade qualifier fields of the incoming messages) against the different qualifiers stored by table 1900.

Each message qualifier is typically a small binary code which signals a condition. Each qualifier in table 1900 preferably maps to an entry in table 1902 which contains a pass/drop flag in each address. Thus, if a match is found any of the message fields with
5 respect to qualifier [1] from table 1900, then this match would map to the entry in position n-2 of table 1902, and the bit value in position n-2 of table 1902 would be retrieved and provided as an input to AND logic 1904. Message qualifiers from table 1900 can be directly addressed to entries in table 1902. However, if the size
10 of table 1900 were to greatly increase, it should be noted that hashing could be used to map qualifiers to pass/drop flags.

If any of the inputs to AND logic 1904 are high, then this will cause the pass/drop signal 1906 to follow as high. Logic 1908 is configured to be controlled by signal 1906 when deciding whether
15 to pass incoming messages to the output. Thus, if any of the matching qualifiers from table 1900 maps to a high bit in table 1902, this will result in the subject message being passed on to the output of the message qualifier filter module 1800.

In this manner, each qualifier in the qualifier table may
20 define a message criteria which, if found in an incoming message, will result in that incoming message being passed along or blocked from propagating downstream in pipeline 500. Examples of message qualifiers which can be stored by table 1900 to indicate that a message should be blocked would be indicators in messages that
25 identify the message as any of a non-firm quote, penalty bid, withdrawn quote, held trade, and out of sequence trade.

Tables 1900 and 1902 are preferably located within available on-chip memory for the reconfigurable logic device 102. However, it should be noted that these tables could alternatively be stored on
30 any memory resource within or accessible to coprocessor 140.

It should also be noted that the incoming messages to pipeline 500 are preferably normalized messages in that the message fields are formatted in a manner expected by the pipeline 500. Such normalization may optionally have been already been performed prior
35 to the time that the messages reach pipeline 500. However, optionally, pipeline 500 may include a message normalization module

2000 as shown in Figure 20 to provide the normalization functions. Such normalization preferably includes functions such as symbol ID mapping (wherein each message is assigned with a symbol ID (preferably a fixed-size binary tag) that uniquely identifies the financial instrument which is the subject of the message) and GEID mapping (wherein each message is assigned with a GEID (preferably also a fixed-size binary tag) that uniquely identifies the exchange corresponding to the message). FAM modules to perform such normalization functions are described in the above-referenced and incorporated U.S. patent application 11/765,306.

With respect to administering the BCE 400, it should be noted that baskets and trigger conditions can be created, modified, and/or deleted in any of a number of ways. For example, a system administrator and/or client application can be given the ability to define baskets and/or trigger conditions

To create and/or modify a basket, appropriate entries and allocations will be needed for the new/modified basket in the various tables described herein in connection with pipeline 500. A system administrator may use a control interface to add/modify basket configurations. Client applications may submit basket definitions via an application programming interface (API) that runs on their local machine(s).

The BCE pipeline may be deployed on coprocessor 140 of a ticker plant platform 2100 as shown in Figure 21. Additional details regarding such a ticker plant platform 2100 can be found in the above-referenced and incorporated 11/765,306 patent application. In summary, the financial market data from the exchanges is received at the O/S supplied protocol stack 2102 executing in the kernel space on processor 112 (see Figures 1(a)-(b)). An upjct driver 2104 delivers this exchange data to multi-threaded feed pre-processing software 2112 executing in the user-space on processor 112. These threads may then communicate data destined for the coprocessor 140 to the hardware interface driver software 2106 running in the kernel space.

Instructions from client applications may also be communicated to the hardware interface driver 2106 for ultimate delivery to

coprocessor 140 to appropriately configure a BCE pipeline that is instantiated on coprocessor 140. Such instructions arrive at an O/S supplied protocol stack 2110 from which they are delivered to a request processing software module 2116. A background and maintenance processing software module 2114 thereafter determines whether the client application has the appropriate entitlement to add/ change/modify a basket. If so entitled, the background and maintenance processing block 2114 communicates a command instruction to the hardware interface driver 2106 for delivery to the coprocessor to appropriately update the table entries in the BCE pipeline to reflect the added/changed/modified basket, as previously explained above. Deletions of baskets as well as additions/modifications/ deletions of trigger conditions from the BCE pipeline can be performed via a like process, as indicated above.

The hardware interface driver 2106 then can deliver an interleaved stream of financial market data and commands to the coprocessor 140 for consumption thereby. Outgoing data from the coprocessor 140 returns to the hardware interface driver 2106, from which it can be supplied to MDC driver 2108 for delivery to the client connections (via protocol stack 2110) and/or delivery to the background and maintenance processing block 2114.

As discussed above, some financial instrument baskets such as ETFs may have some form of cash component to them. These cash components can be taken into account by the system in any of a number of ways.

For example, to insert a cash component into a basket's NAV, a control process can be configured to generate a synthetic event for delivery to the pipeline. A symbol ID would be assigned to a given cash account, and the price for that synthetic symbol ID would be \$1 and the weight value w for that cash account would be set equal to the cash value of the cash account. The tables used by the BAL module would be updated to reflect this synthetic symbol ID and cause the cash account to impact the NAV for the basket. In a more complex scenario, events could be delivered to the pipeline so that

the value of the cash account can fluctuate with the value of the pertinent currency on the foreign exchange market.

Another way of injecting a cash component of a basket into the system is to define the trigger threshold values in tables 1314, 1316, 1318, 1320, and 1322 such that the cash component amounts are built into those trigger threshold values.

Also, it should be noted that the BAL module (or the BVU module) can be configured to drop events where all of the delta prices for that event are zero.

Similarly, it should be noted that in the embodiment of Figure 9, the NAV update engine 950 is configured to compute updated NAVs even for NAV types whose corresponding delta prices are zero. That is, a current delta event being processed by the BVU module may contain a \$0.03 value for the Δ Bid price but zero values for the Δ Ask and Δ Last prices. With the embodiment of Figure 9, some of the different NAV compute logic engines 916 of the parallel paths would thus be used to compute a new NAV where there will not be a change relative to the old NAV (e.g., for the example given, the Ask NAV^{new}, Last NAV^{new}, and Ask+Tick NAV^{new} values would be unchanged relative to their old values). It is believed that this configuration will still yield effective efficiency given that most of the events processed by the pipeline are expected to comprise quote activity. However, a practitioner is free to choose to increase the efficiency of the BVU module by not including any zero delta prices in the delta events and including a dynamic scheduler and memory controller which directs delta events to the appropriate computing path(s) within the NAV update engine 950 to more fully utilize all available computing paths within the NAV update engine 950.

It should also be noted that logic could be added to the BVU modules such that the NAV computations are also based on derived statistics such as the Volume Weighted Average Price (VWAP) and use other fields which may be available in a message, such as the difference in traded volume.

It should also be noted that the coprocessor 140 can be configured to perform computations in addition to those of the BCE if desired by a practitioner of an embodiment of the invention. For

example, the coprocessor 140 can also employ a last value caching (LVC) module, as described in the above-referenced and incorporated 11/765,306 patent application.

It should be noted that a practitioner of an embodiment of the invention may choose to compute the value for Δj but not the value for NAV^{new} , in which case the BVU module need not produce any NAV^{new} values. In such instances, the trigger conditions used by the PET module 1200 can be built around variation in Δj rather than NAV^{new} . In such a situation, the client could optionally determine the NAV values itself from the Δj value, although a practitioner may find value in the Δj values themselves apart from their actual combination with an old NAV value to find a new NAV value.

Also, while the preferred embodiment for the BAL module 502 is configured to compute the delta values for the last/bid/ask prices in the incoming messages, it should be noted that the computation of such delta values can optionally be performed by one or more modules upstream from the BAL module 502 or the BCE pipeline 500 altogether. As such, the input messages to the BCE pipeline 500 may already contain the ΔBid , ΔAsk , and/or $\Delta Last$ prices for the subject financial instruments. In such instances, table 608 would not need to store the previous bid/ask/last prices for each financial instrument, and BAL module 502 need not employ the logic necessary to compute those delta prices.

Furthermore, while in the preferred embodiment disclosed herein the coprocessor 140 comprises a reconfigurable logic device 102 such as an FPGA, it should be noted that the coprocessor 140 can be realized using other processing devices. For example, the coprocessor 140 may comprise graphics processor units (GPUs), general purpose graphics processors, chip multi-processors (CMPs), dedicated memory devices, complex programmable logic devices, application specific integrated circuits (ASICs), and other I/O processing components. Moreover, it should be noted that system 100 may employ a plurality of coprocessors 140 in either or both of a sequential and a parallel multi-coprocessor architecture.

Further still, while the inventors believe that special advantages exist in connection with using the pipeline disclosed

herein to process financial information and compute updated values for financial instrument baskets, the inventors further note that the disclosed pipeline also provides latency advantages when processing non-financial data which pertains to baskets. For example, the basket calculation engine may be configured to compute the net asset value of the inventory held by a multi-national retail organization. Incoming events to the pipeline that correspond to product sales and/or product deliveries could thus be used to update stored values for each product, wherein these stored values correspond to inventory count values. A weight assigned to each product would correspond to a price value assigned to the product. Thus, a pipeline such as the one disclosed herein could be used to closely track the net inventory values of the organization's products. Triggers may then be used to notify management when inventory values swell or shrink by a given threshold.

Another example would be for a basket which is a collection of data points from some type of scientific experiment. Each data point may have associated values such as size, mass, etc., and these data points may arrive over time as streaming data from one or more monitoring stations. The pipeline can readily be configured to derive a size NV by computing a weighted sum of the sizes from the incoming data stream and/or a mass NV by computing a weighted sum of the masses from the incoming data stream.

While the present invention has been described above in relation to its preferred embodiments, various modifications may be made thereto that still fall within the invention's scope. Such modifications to the invention will be recognizable upon review of the teachings herein. Accordingly, the full scope of the present invention is to be defined solely by the appended claims and their legal equivalents.

WHAT IS CLAIMED IS:

1. A method of processing data, the method comprising:
streaming data through a processor, the data comprising a
message, the message corresponding to a value for an element;
5 retrieving stored data based on the message, the retrieved
data corresponding to a basket, the basket having the element as a
member; and
computing a net value for the basket based on the retrieved
data and the message;
10 wherein the retrieving step and the computing step are
performed by the processor as the data streams through the
processor.
2. The method of claim 1 wherein the processor comprises a
15 coprocessor.
3. The method as defined in any of claims 1-2 wherein the data
comprises financial market data, and wherein the element comprises a
financial instrument.
20
4. The method as defined in any of claims 1-3 further comprising
performing the retrieving step and the computing step in a pipelined
manner.
- 25 5. The method as defined in any of claims 1-4 wherein the
computed net value comprises a net asset value (NAV) for the basket.
6. The method as defined in any of claims 1-5 wherein the message
comprises price information about the financial instrument, and
30 wherein the computing step comprises computing the NAV using a delta
calculation approach that is based on a contribution of the
financial instrument price information to the NAV.
7. The method as defined in any of claims 1-6 wherein the
35 retrieving step further comprises retrieving a previous value
relating to the basket and wherein the NAV computing step comprises

adding the contribution to the retrieved previous value to thereby compute the NAV.

5 8. The method as defined in any of claims 1-7 wherein the previous value comprises a previous NAV for the basket.

9. The method as defined in any of claims 1-8 wherein the previous value comprises S, wherein S represents a value corresponding to a previous NAV for the basket multiplied by a
10 divisor value for the basket, and wherein the NAV computing step further comprises dividing the sum of the contribution and S by the divisor value.

10. The method as defined in any of claims 1-9 wherein the NAV
15 computing step comprises simultaneously computing a plurality of different NAVs for the basket, each of the NAVs corresponding to a different NAV type.

11. The method as defined in any of claims 1-10 wherein the
20 plurality of different NAV types comprise a bid NAV, an ask NAV, and a last NAV.

12. The method as defined in any of claims 1-11 wherein the
25 plurality of different NAV types further comprise a bid-tick NAV and an ask+tick NAV.

13. The method as defined in any of claims 1-12 wherein the financial instrument is a member of a plurality of baskets, wherein the retrieved data corresponds to the baskets, and wherein the
30 computing step comprises computing the NAV for each of the baskets.

14. The method as defined in any of claims 1-13 wherein the computing step comprises computing the contribution to the NAV for each basket based on the financial instrument price information and
35 a weight value assigned to the financial instrument for that basket,

and wherein the retrieving step further comprises retrieving the weight values for the baskets.

15. The method as defined in any of claims 1-14 wherein the
5 retrieving step further comprises retrieving a basket identifier for each of the baskets, each basket identifier corresponding to a different one of the baskets and being associated with a different one of the weight values.

10 16. The method as defined in any of claims 1-15 wherein the NAV computing step comprises calculating a price delta for the financial instrument, the method further comprising generating a plurality of delta events based on the retrieved data, each delta event
15 comprising the calculated price delta, a retrieved basket identifier, and a retrieved weight value, and wherein the NAV computing step comprises computing the NAVs based on each delta event, and wherein the retrieving step, the generating step, and the NAV computing step are performed by the coprocessor in a pipelined manner as the financial market data streams through the coprocessor.

20 17. The method as defined in any of claims 1-16 further comprising:

storing the basket identifiers and their associated weight values in a table as plurality of pairs.

25 18. The method as defined in any of claims 1-17 wherein the storing step further comprises storing previous price information for the financial instrument in the table.

30 19. The method as defined in any of claims 1-18 further comprising updating the previous price information in the table based on the message.

35 20. The method as defined in any of claims 1-19 wherein the message comprises a financial instrument identifier for the financial instrument corresponding thereto, wherein the table

comprises a first table, wherein the storing step further comprises storing a plurality of pointers in a second table, the pointers being indexed by a financial instrument identifier, and wherein the pairs in the first table are indexed by the pointers, and wherein
5 the retrieving step further comprises (i) retrieving a pointer from the second table based on the message's financial instrument identifier, and (ii) retrieving the pairs from the first table based on the retrieved pointer.

10 21. The method as defined in any of claims 1-20 further comprising adding and deleting basket definitions within the tables in response to input from a client.

15 22. The method as defined in any of claims 1-21 further comprising performing at least one of the group consisting of (i) adding a financial instrument to, (ii) modifying a weight for a financial instrument with respect to, and (iii) deleting a financial instrument from a basket by modifying the tables in response to input from a client.

20

23. The method as defined in any of claims 1-22 wherein the message comprises a global exchange identifier, and wherein the retrieving step comprises retrieving data corresponding to a composite financial instrument and data corresponding to at least
25 one regional financial instrument based on the global exchange identifier.

24. The method as defined in any of claims 1-23 further comprising:

30 processing the NAV against a triggering condition to determine whether the NAV is to be reported to a client; and

generating a message for delivery to the client in response to a determination by the processing step that the NAV is to be reported to the client, the generated message comprising the NAV.

35

25. The method as defined in any of claims 1-24 wherein the retrieving step, the computing step, and the processing step are performed by the coprocessor as the financial market data streams through the coprocessor.

5

26. The method as defined in any of claims 1-25 further comprising performing the retrieving step, the computing step, and the processing step in a pipelined manner.

10

27. The method as defined in any of claims 1-26 wherein the retrieving step, the computing step, the processing step, and the generating step are performed in a pipelined manner by the coprocessor as the financial market data streams through the coprocessor.

15

28. The method as defined in any of claims 1-27 wherein the processing step comprises:

retrieving a reference value for the basket from a first table based on a basket identifier associated with the computed NAV;

20

determining a dissimilarity between the retrieved reference value and the computed NAV;

retrieving a trigger threshold value for the basket from a second table based on the basket identifier; and

25

comparing the dissimilarity with the retrieved trigger threshold value to thereby determine whether the NAV is to be reported to the client.

30

29. The method as defined in any of claims 1-28 wherein the dissimilarity determining step comprises computing a difference between the retrieved reference value and the computed NAV.

30. The method as defined in any of claims 1-29 further comprising defining a plurality of triggering conditions for a plurality of baskets in response to input from the client.

35

31. The method as defined in any of claims 1-30 wherein the financial market data comprises a plurality of messages, the method further comprising:

filtering the messages prior to the retrieving step based on
5 at least one filtering criterion such that the retrieving step is performed on the messages which pass the filtering step.

32. The method as defined in any of claims 1-31 wherein the filtering step, the retrieving step, and the computing step are
10 performed by the coprocessor as the financial market data streams through the coprocessor.

33. The method as defined in any of claims 1-32 further comprising performing the filtering step, the retrieving step, and the
15 computing step in a pipelined manner.

34. The method as defined in any of claims 1-33 wherein the coprocessor comprises a reconfigurable logic device, and wherein the performing step comprises performing the retrieving step and the
20 computing step with the reconfigurable logic device.

35. A system for processing data, the system comprising:
a processor configured to (i) receive a stream of data, the data comprising at least one message, the message corresponding to a
25 value for an element, (ii) retrieve stored data based on the message, the retrieved data corresponding to a basket, the basket having the element as a member, and (iii) compute a net value for the basket based on the retrieved data and the message.

36. The system of claim 35 wherein the processor comprises a
30 coprocessor.

37. The system as defined in any of claims 35-36 wherein the processor further comprises a main processor in communication with
35 the coprocessor, the processor being configured to deliver the stream of data to the coprocessor.

38. The system as defined in any of claims 35-37 wherein the data comprises financial market data, and wherein the element comprises a financial instrument.

5

39. The system as defined in any of claims 35-38 wherein the coprocessor comprises a basket association lookup module and an basket value updating module arranged in a pipelined manner, wherein the basket association lookup module is configured to retrieve the
10 stored data based on the message, and wherein the basket value updating module is configured to compute the net value for the basket based on the retrieved data and the message.

40. The system as defined in any of claims 35-39 wherein the
15 computed net value comprises a net asset value (NAV) for the basket, wherein the message comprises price information about the financial instrument, and wherein the basket value updating module is further configured to compute the NAV using a delta calculation approach that is based on a contribution of the financial instrument price
20 information to the NAV.

41. The system as defined in any of claims 35-40 wherein the basket value updating module is further configured to simultaneously compute a plurality of different NAVs for the basket, each of the
25 NAVs corresponding to a different NAV type.

42. The system as defined in any of claims 35-41 wherein the financial instrument is a member of a plurality of baskets, wherein the retrieved data corresponds to the baskets, and wherein the
30 basket value updating module is further configured to compute the NAV for each of the baskets.

43. The system as defined in any of claims 35-42 wherein the basket value updating module is further configured to compute the
35 contribution to the NAV for each basket based on the financial instrument price information and a weight value assigned to the

financial instrument for that basket, and wherein the basket association lookup module is further configured to retrieve the weight values for the baskets.

5 44. The system as defined in any of claims 35-43 wherein the basket association lookup module is further configured to retrieve a basket identifier for each of the baskets, each basket identifier corresponding to a different one of the baskets and being associated with a different one of the weight values.

10

45. The system as defined in any of claims 35-44 wherein the message comprises a global exchange identifier, and wherein the basket association lookup module is further configured to retrieve data corresponding to a composite financial instrument and data
15 corresponding to at least one regional financial instrument based on the global exchange identifier.

46. The system as defined in any of claims 35-45 wherein the coprocessor further comprises a price event trigger module in
20 communication with the basket value updating module, the price event trigger module being configured to process the NAV against a triggering condition to determine whether the NAV is to be reported to a client.

25 47. The system as defined in any of claims 35-46 wherein the basket association lookup module, the basket value updating module, and the price event trigger module are arranged in a pipelined manner.

30 48. The system as defined in any of claims 35-47 wherein the coprocessor further comprises an event generator module in communication with the price event trigger module, wherein the event generator module is configured to generate a message for delivery to the client in response to a determination by the price event trigger
35 module that the NAV is to be reported to the client, the generated message comprising the NAV.

49. The system as defined in any of claims 35-48 wherein the basket association lookup module, the basket value updating module, and the price event trigger module are arranged in a pipelined
5 manner.

50. The system as defined in any of claims 35-49 wherein the price event trigger module is further configured to:

retrieve a reference value for the basket from a first table
10 based on a basket identifier associated with the computed NAV;
determine a dissimilarity between the retrieved reference value and the computed NAV;

retrieve a trigger threshold value for the basket from a second table based on the basket identifier; and

15 compare the dissimilarity with the retrieved trigger threshold value to thereby determine whether the NAV is to be reported to the client.

51. The system as defined in any of claims 35-50 wherein the
20 financial market data comprises a plurality of messages, and wherein the coprocessor further comprises a message qualifier filter in communication with the basket association lookup module, the message qualifier filter being configured to filter the messages based on at least one filtering criterion such that the basket association
25 lookup module receives the messages which pass the message qualifier filter.

52. The system as defined in any of claims 35-51 wherein the message qualifier filter, the basket association lookup module, and
30 the basket value updating module are arranged in pipelined manner.

53. The system as defined in any of claims 35-52 wherein the coprocessor comprises a reconfigurable logic device, the reconfigurable logic device comprising the basket association lookup
35 module and the basket value updating module.

54. The system as defined in any of claims 35-53 further comprising another processor in communication with the processor, wherein the another processor is configured to execute a client application, and wherein the main processor is further configured to deliver data corresponding to the computed basket value to the client application.

55. A method of processing data, the method comprising:
performing a plurality of basket calculation operations on a bit stream using a processor as the bit stream streams through the processor to thereby compute a plurality of net basket values.

56. The method of claim 55 wherein the bit stream is representative of financial market data, and wherein the net basket values comprise a plurality of net asset values (NAVs) for financial instrument baskets.

57. The method as defined in any of claims 55-56 wherein the processor comprises a coprocessor and a main processor, the main processor being configured to deliver the bit stream to the coprocessor, and the coprocessor being configured to perform the basket calculation operations.

58. The method as defined in any of claims 55-57 wherein the performing step further comprises performing the basket calculation operations via a plurality of pipelined modules which are deployed on the coprocessor such that each pipeline module operates on different financial market data within the bit stream at the same time.

59. The method as defined in any of claims 55-58 wherein the coprocessor comprises a reconfigurable logic device.

60. The method as defined in any of claims 55-59 wherein the performing step further comprises:

processing the bit stream to determine which data should be
 retrieved from a plurality of tables;
 retrieving the determined data from the tables; and
 computing the basket NAVs based on the retrieved data and the
 5 bit stream.

61. An apparatus for computing a value pertaining to a net asset
 value (NAV) for a basket, the apparatus comprising:

a firmware pipeline configured to (i) receive a message
 10 comprising price information for a financial instrument and an
 identifier for the financial instrument, (ii) determine a plurality
 of baskets in which the financial instrument is a member based at
 least in part on the financial instrument identifier, and (iii)
 compute a delta contribution to a new NAV for each of the plurality
 15 of baskets.

62. The apparatus of claim 61 wherein the firmware pipeline is
 further configured to compute the new NAV for each of the plurality
 of baskets based on the computed delta contribution.

20

63. The apparatus as defined in any of claims 61-62 wherein the
 firmware pipeline is further configured to (i) compute the delta
 contribution to the new NAV for each of the plurality of baskets
 according to the formula $\Delta j = \frac{w_j}{d} (T_j^{new} - T_j^{old})$, wherein w_j represents a

25

weight for the financial instrument in the basket, wherein d
 represents a divisor value for the basket, wherein T_j^{new} represents
 the price information, and wherein T_j^{old} represents old price
 information for the financial instrument, and (ii) compute the new
 NAV for each of the plurality of baskets according to the formula

30

$NAV^{new} = NAV^{old} + \Delta j$, wherein NAV^{new} is the new NAV for a basket,
 wherein NAV^{old} is an old NAV for a basket.

64. The apparatus as defined in any of claims 61-63 further
 comprising a plurality of tables in which the w_j values, the d

values, the T_j^{old} values, and the NAV^{old} values are stored, and wherein the firmware pipeline is further configured to retrieve the w_j values, the d values, the T_j^{old} values, and the NAV^{old} values from the tables as the message streams through the pipeline.

5

65. The apparatus as defined in any of claims 61-64 wherein the firmware pipeline is implemented on a coprocessor.

66. The apparatus as defined in any of claims 61-65 wherein the
10 coprocessor comprises a reconfigurable logic device.

67. The apparatus as defined in any of claims 61-66 wherein the tables are stored within the reconfigurable logic device.

15 68. The apparatus as defined in any of claims 61-67 wherein the firmware pipeline is further configured to (i) compute the delta contribution to the new NAV for each of the plurality of baskets according to the formula $w_j(T_j^{new} - T_j^{old})$, wherein w_j represents a weight for the financial instrument in the basket, wherein T_j^{new} represents
20 the price information, and wherein T_j^{old} represents old price information for the financial instrument, (ii) retrieve a value S^{old} for each of the baskets, (iii) compute a sum of S^{old} and $w_j(T_j^{new} - T_j^{old})$ for each of the baskets, (iv) retrieve a divisor value for the baskets, and (v) compute the new NAV for each of the plurality of
25 baskets by dividing each sum by the retrieved divisor value.

69. A method for processing financial market data, the method comprising:

processing a stream of financial market data using a
30 coprocessor to compute a net asset value for a financial instrument basket as the financial market data streams through the coprocessor; and

detecting an arbitrage condition in connection with the basket based on the computed net asset value.

70. The method of claim 69 further comprising:

placing at least one trade order with at least one financial
market in response to the detecting step to thereby take advantage
5 of the detected arbitrage condition.

71. The method as defined in any of claims 69-70 wherein the
detecting step comprises detecting the arbitrage condition using the
coprocessor as the financial market data streams through the
10 coprocessor.

72. The method as defined in any of claims 69-71 wherein the
detecting step further comprises:

computing a difference between the computed net asset value
15 and a reference value; and

comparing the computed difference with a trigger threshold to
detect whether the arbitrage condition exists.

73. The method as defined in any of claims 69-72 wherein the
20 coprocessor comprises a reconfigurable logic device.

74. The method as defined in any of claims 69-73 wherein the
basket comprises an exchange traded fund (ETF).

25 75. A method for processing data, the method comprising:

receiving a stream of data, the data comprising a value for an
element;

determining a basket which pertains to the element; and

calculating a value applicable to the determined basket based
30 at least in part on the element value; and

wherein the determining step is performed by a first module in
a pipeline; and

wherein the calculating step is performed by a second module
in the pipeline.

35

76. The method of claim 75 wherein the data comprises financial market data, wherein the element comprises a financial instrument, wherein the element value comprises price information for the financial instrument, and wherein the basket comprises a financial instrument basket.

77. The method as defined in any of claims 75-76 wherein the calculated value comprises a net asset value for the financial instrument basket, the method further comprising detecting an arbitrage condition in connection with the basket based on the calculated net asset value.

78. The method as defined in any of claims 75-77 further comprising:

placing at least one trade order with at least one financial market in response to the detecting step to thereby take advantage of the detected arbitrage condition.

79. The method as defined in any of claims 75-78 wherein the detecting step is performed by a third module in the pipeline.

80. The method as defined in any of claims 75-79 wherein the detecting step further comprises:

computing a difference between the calculated net asset value and a reference value; and

comparing the computed difference with a trigger threshold to detect whether the arbitrage condition exists.

81. The method as defined in any of claims 75-80 further comprising defining the reference value and the trigger threshold in response to client input.

82. The method as defined in any of claims 75-81 wherein the calculating step comprises computing a delta contribution of the financial instrument to the determined basket's net asset value.

83. The method as defined in any of claims 75-82 wherein the calculating step further comprises computing the determined basket's net asset value based on the computed delta contribution.

5 84. The method as defined in any of claims 75-83 wherein the pipeline comprises a firmware pipeline.

85. The method as defined in any of claims 75-84 wherein the firmware pipeline is deployed in reconfigurable logic.

10

86. The method as defined in any of claims 75-85 wherein the basket comprises an exchange traded fund (ETF).

87. The method as defined in any of claims 75-86 wherein the
15 calculating step comprises computing a delta contribution of the financial instrument to a net asset value for the determined basket.

88. A method of administering a plurality of tables corresponding to a plurality of baskets, the method comprising:

20 storing a plurality of pointers in a first table, each pointer being indexed by an identifier for a financial instrument;

storing a plurality of header blocks in a second table, the header blocks being indexed by the pointers, each header block being associated with a financial instrument, each header block comprising
25 a plurality of basket identifiers for the baskets and a plurality of weight values corresponding to the baskets, the basket identifiers and weight values being stored in the table as a plurality of pairs, each weight value defining a weight for a financial instrument in the basket corresponding to the basket identifier that is paired
30 with that weight value.

89. The method of claim 88 wherein the header blocks further comprise price information for the financial instruments.

90. The method as defined in any of claims 88-89 wherein each header block further comprises a count field which includes a count value indicative of how many pairs are present in that header block.

5 91. The method as defined in any of claims 88-90 further comprising storing a plurality of extension blocks in the second table, wherein at least a plurality of header blocks further comprise an extension pointer field which includes a pointer to one of the extension blocks, the extension blocks comprising overflow
10 pairs of basket identifiers and weight values.

92. The method as defined in any of claims 88-91 wherein the first table further comprises a plurality of global exchange identifier indexes, each global exchange identifier index corresponding to a
15 different header block in the second table.

93. A method for computing a basket value, the method comprising:
receiving a delta event, the delta event comprising a basket identifier for a basket, a price delta with respect to a financial
20 instrument, the financial instrument being a member of the basket, and a weight value that represents a weight of the financial instrument within the basket; and

retrieving a divisor value for the basket based on the basket identifier;
25 computing a net asset value (NAV) for the basket according to a delta calculation approach based on the delta event and the divisor value.

94. The method of claim 93 further comprising performing the
30 method steps using a coprocessor.

95. An apparatus for computing a basket value, the apparatus comprising:
a processor configured to (i) receive a delta event, the delta
35 event comprising a basket identifier for a basket, a price delta with respect to a financial instrument, the financial instrument

being a member of the basket, and a weight value that represents a weight of the financial instrument within the basket, (ii) retrieve a divisor value for the basket based on the basket identifier, and (iii) compute a net asset value (NAV) for the basket according to a delta calculation approach based on the delta event and the divisor value.

96. A basket association method, the method comprising:

receiving a bit stream, the bit stream being representative of a message pertaining to a financial instrument, the message comprising a symbol identifier for the financial instrument and a global exchange identifier (GEID) for an exchange which pertains to the message;

performing a lookup in a first table based on the symbol ID to retrieve a record from the first table;

determining a pointer from the retrieved record;

determining an index value from the retrieved record based on the GEID;

performing a lookup in the second table based on the pointer and the index value to retrieve basket association data pertaining to each basket of which the financial instrument is a member; and

outputting a bit stream that is representative of the retrieved basket association data.

97. A basket association apparatus, the apparatus comprising:

a processor configured to (i) receive a message pertaining to a financial instrument, the message comprising a symbol identifier for the financial instrument and a global exchange identifier (GEID) for an exchange which pertains to the message, (ii) perform a lookup in a first table based on the symbol ID to retrieve a record from the first table, (iii) determine a pointer from the retrieved record, (iv) determine an index value from the retrieved record based on the GEID, (v) perform a lookup in the second table based on the pointer and the index value to retrieve basket association data pertaining to each basket of which the financial instrument is a member, and (vi) output the retrieved basket association data.

98. A method of reporting updated basket values to a client, the method comprising:

receiving a bit stream, the bit stream being representative of
5 a basket event, the basket event comprising a basket identifier for a basket and an updated net asset value (NAV) for the basket;

retrieving a reference value for the basket based on the
basket identifier;

determining a dissimilarity between the retrieved reference
10 value and the updated NAV;

retrieving a trigger threshold value for the basket based on
the basket identifier;

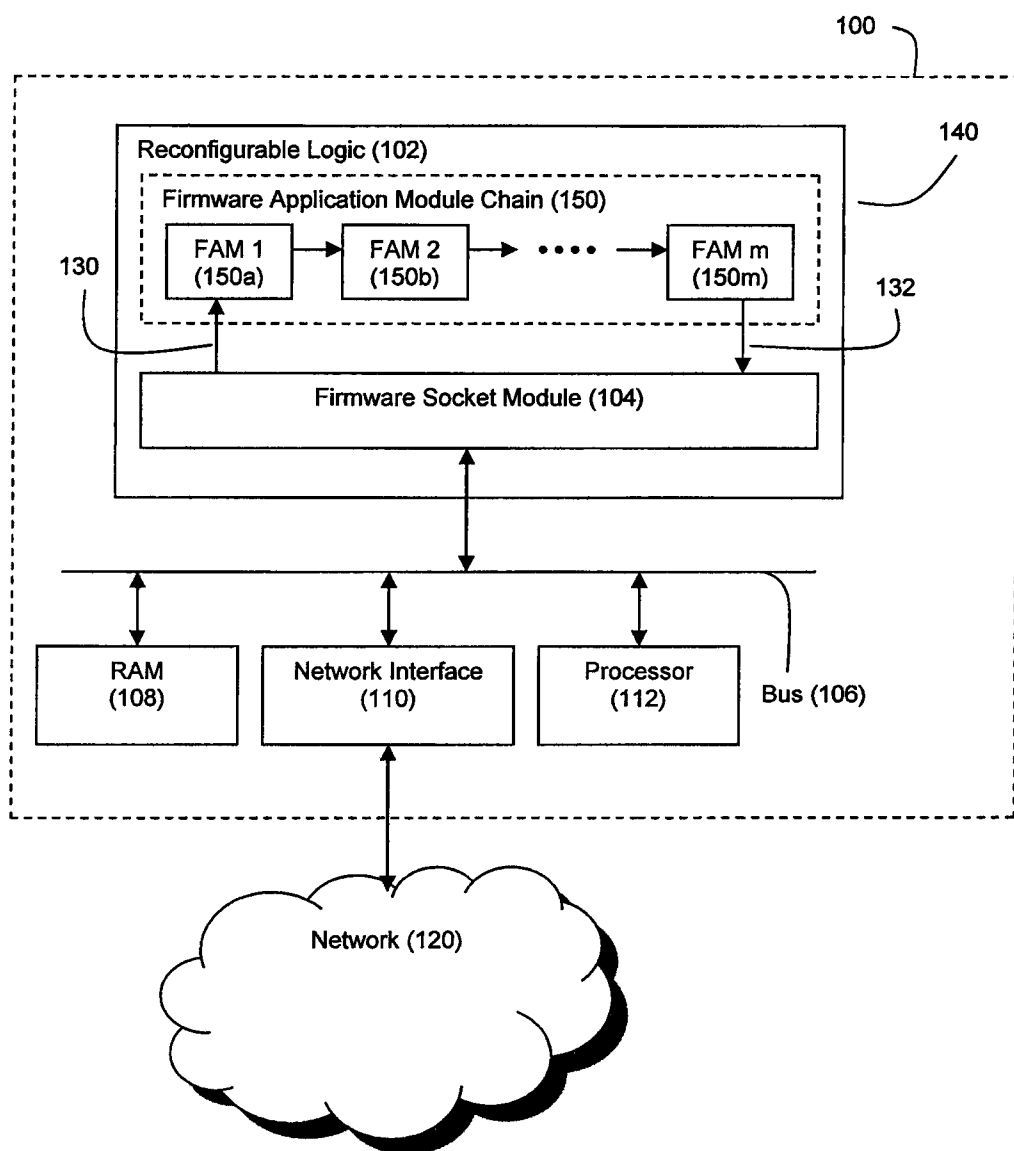
comparing the dissimilarity with the retrieved trigger
threshold value to thereby determine whether the updated NAV is to
15 be reported to the client; and

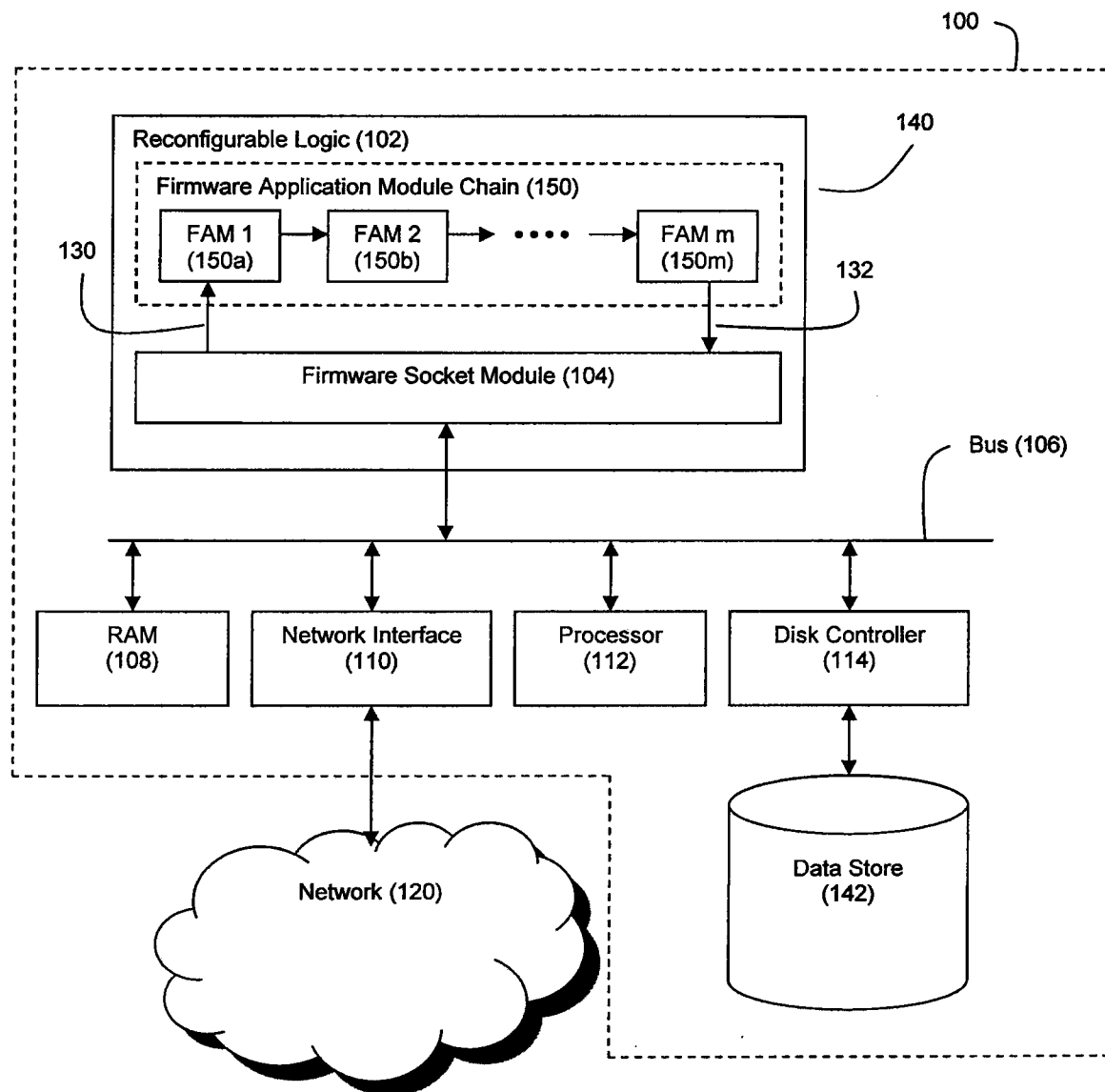
delivering a bit stream that is representative of the updated
NAV to the client in response to a determination by the comparing
step that the updated NAV should be reported to the client.

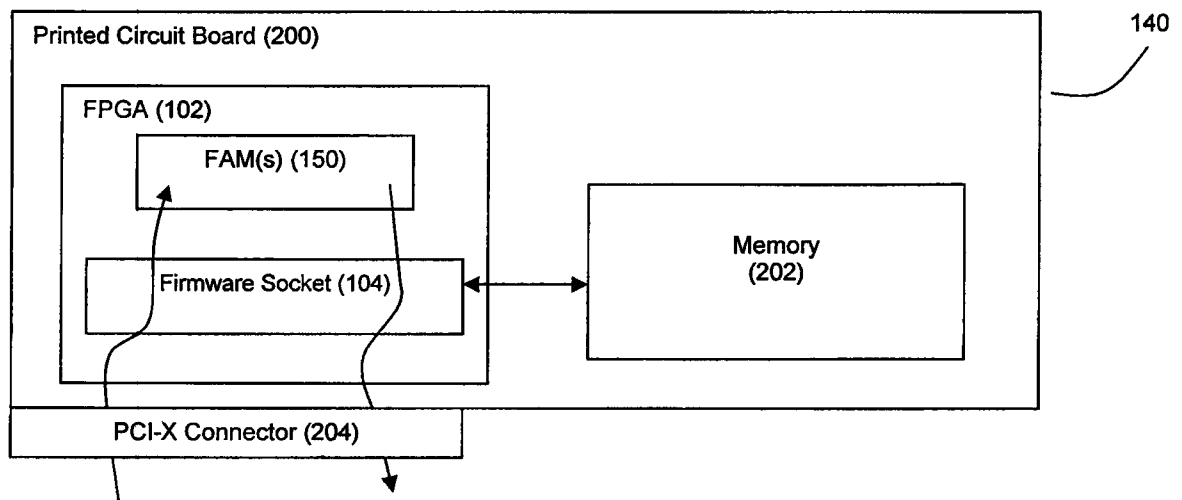
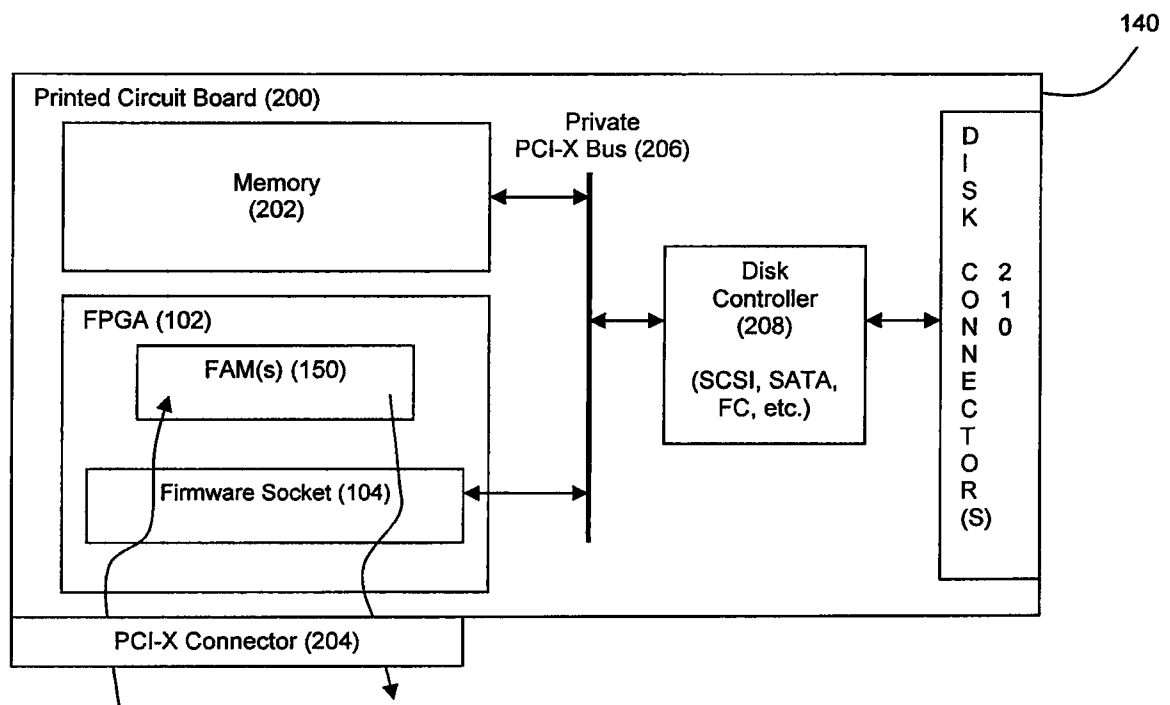
20 99. An apparatus for reporting updated basket values to a client, the apparatus comprising:

a processor configured to (i) receive a basket event, the
basket event comprising a basket identifier for a basket and an
updated net asset value (NAV) for the basket, (ii) retrieve a
25 reference value for the basket based on the basket identifier, (iii)
determine a dissimilarity between the retrieved reference value and
the updated NAV, (iv) retrieve a trigger threshold value for the
basket based on the basket identifier, (v) compare the dissimilarity
with the retrieved trigger threshold value to thereby determine
30 whether the updated NAV is to be reported to the client, and (vi)
report the updated NAV to the client in response to a determination
that the updated NAV should be reported to the client.

1/14

**Figure 1(a)**

**Figure 1(b)**

**Figure 2(a)****Figure 2(b)**

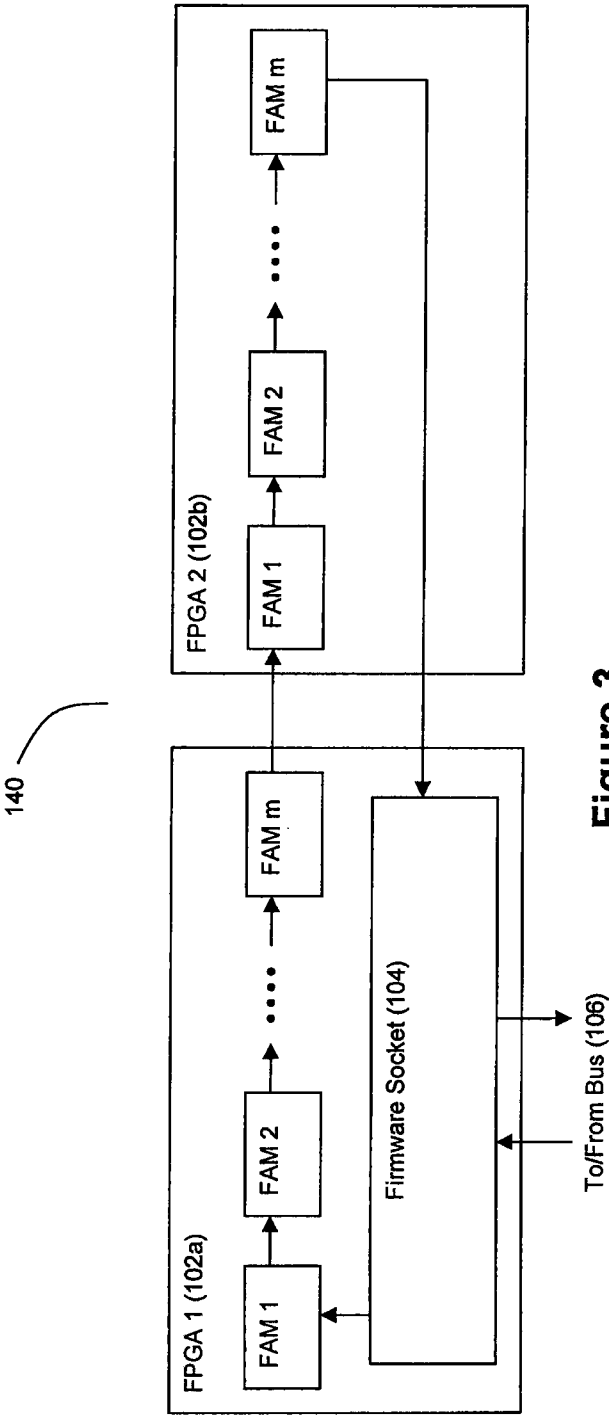


Figure 3

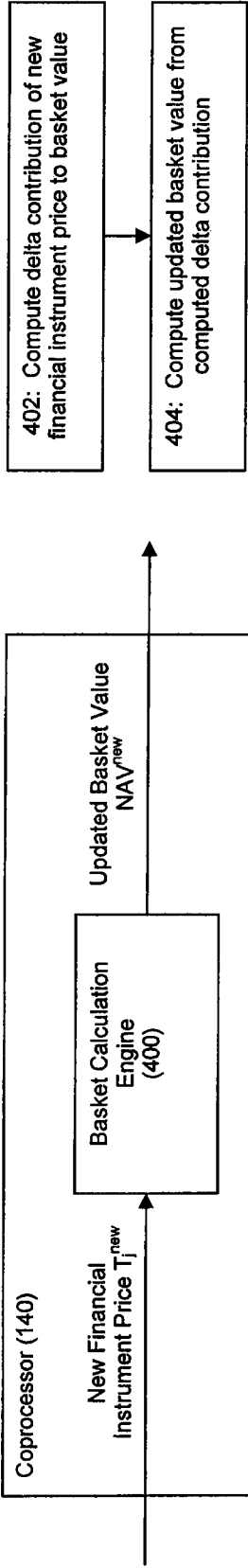


Figure 4(a)

Figure 4(b)

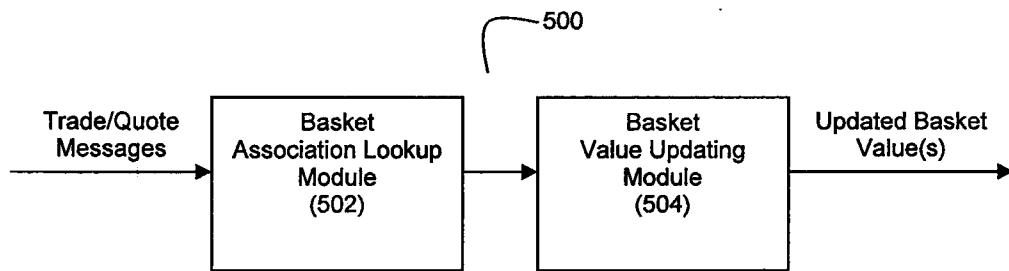


Figure 5

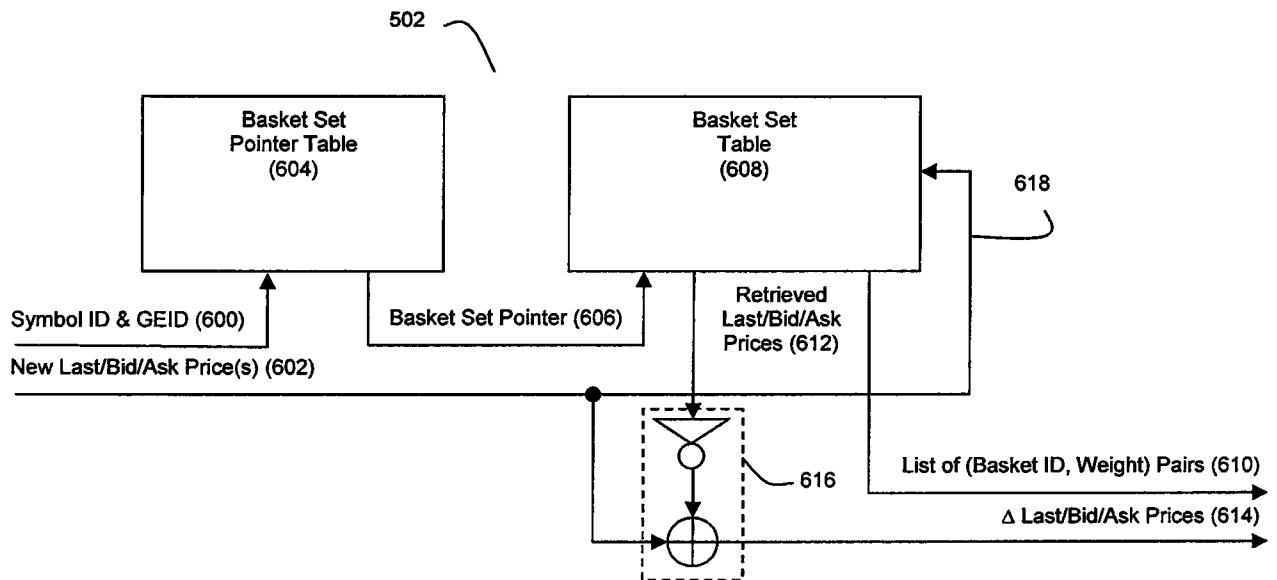


Figure 6

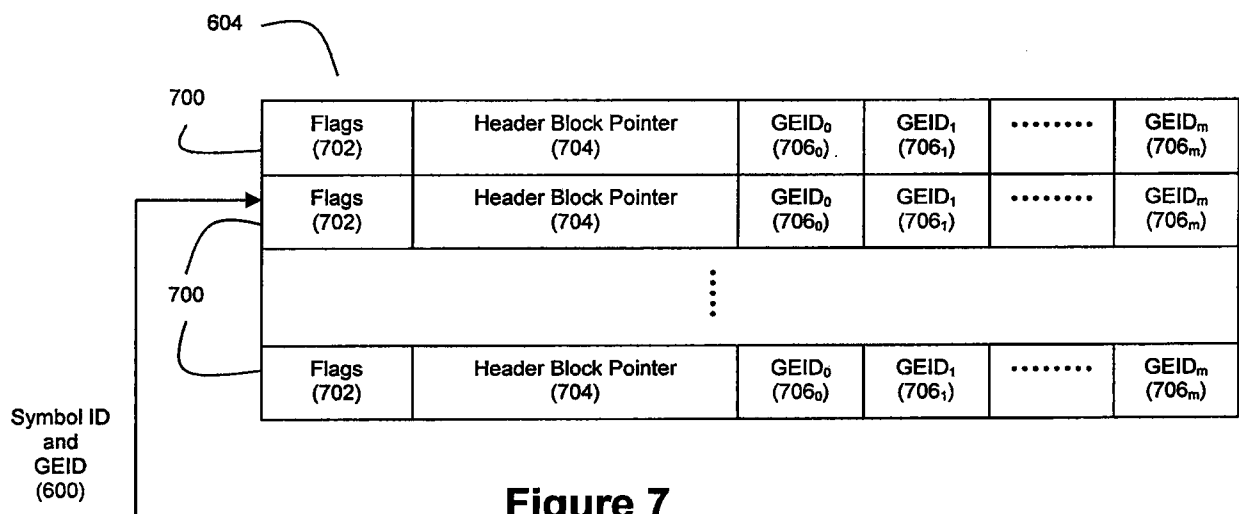


Figure 7

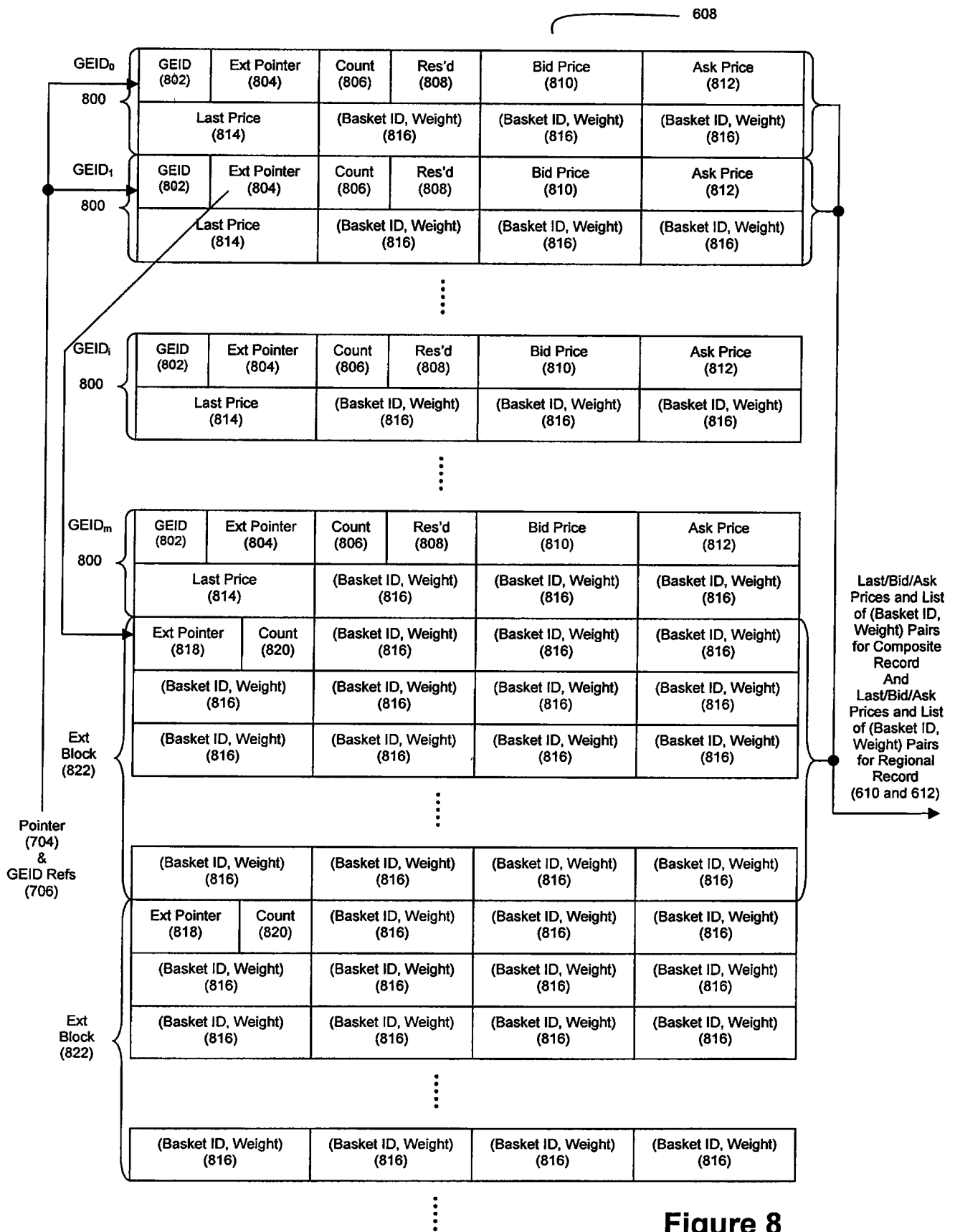


Figure 8

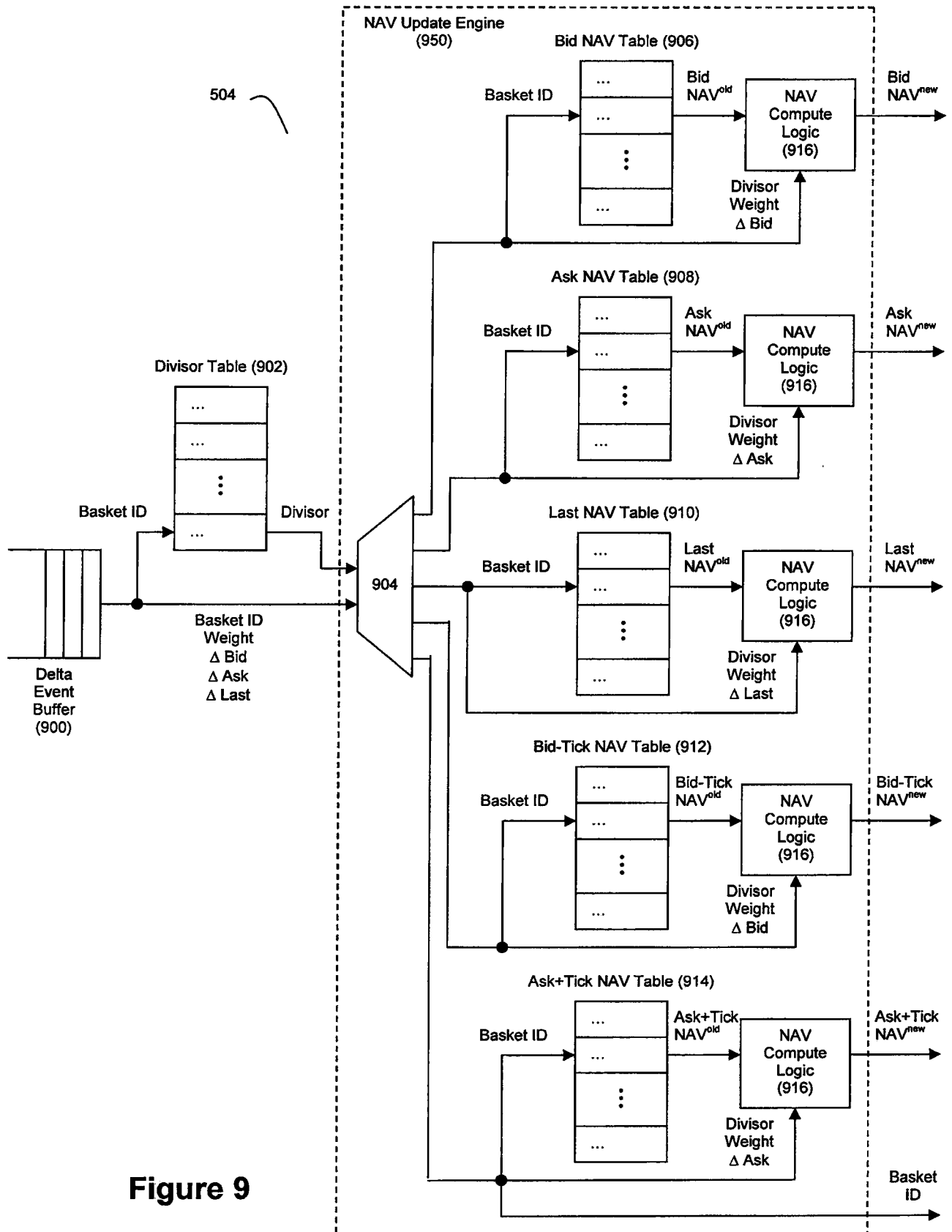


Figure 9

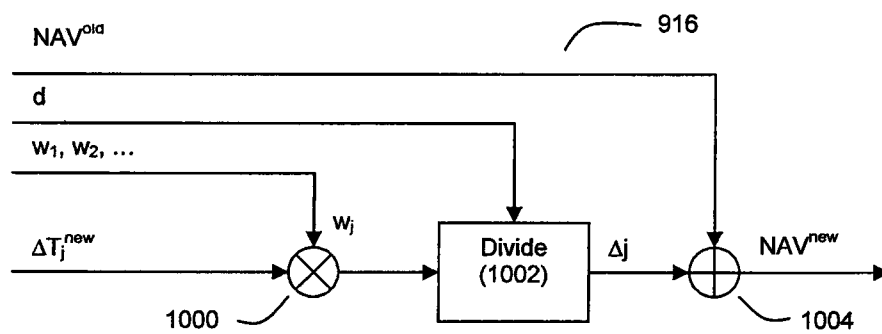


Figure 10(a)

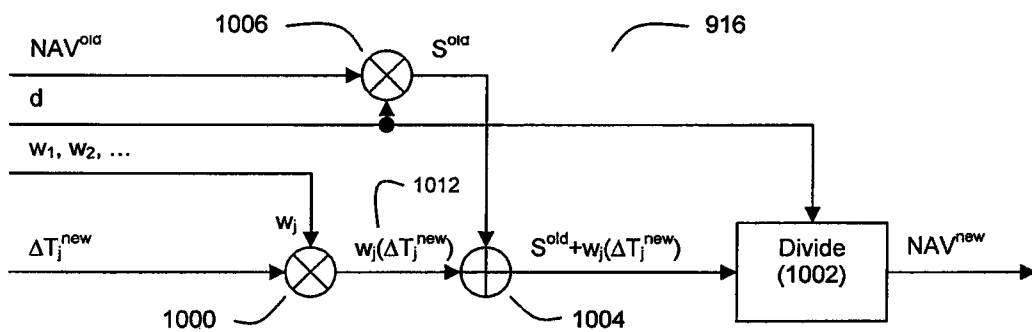


Figure 10(b)

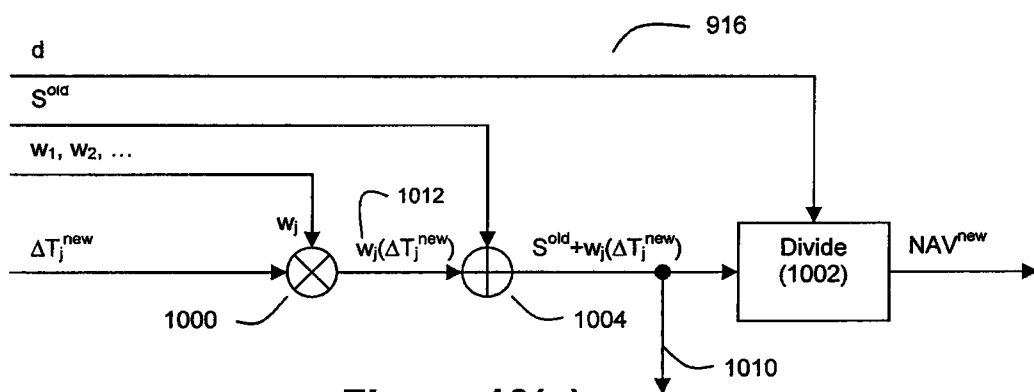


Figure 10(c)

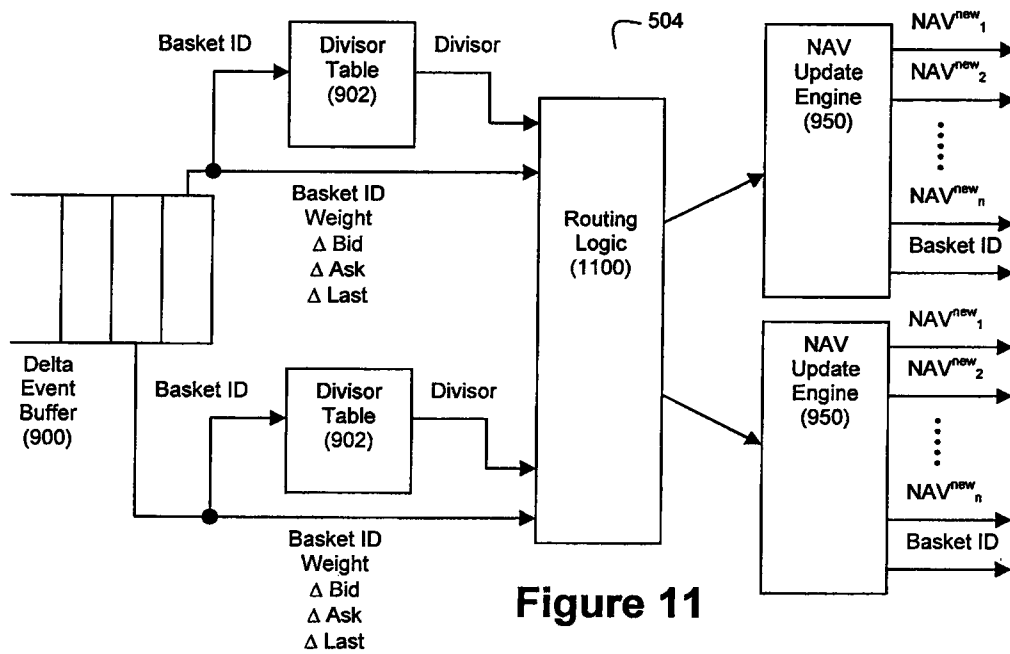


Figure 11

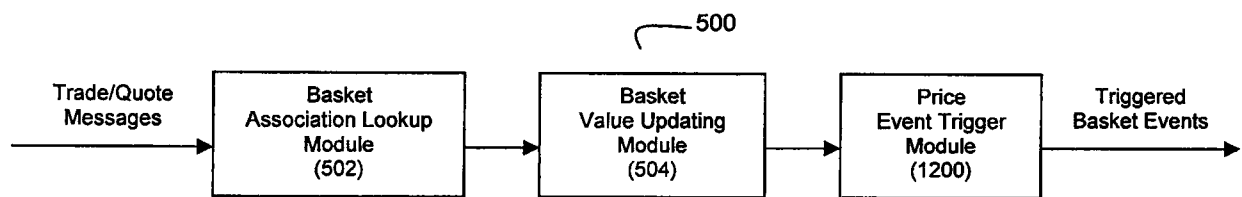


Figure 12

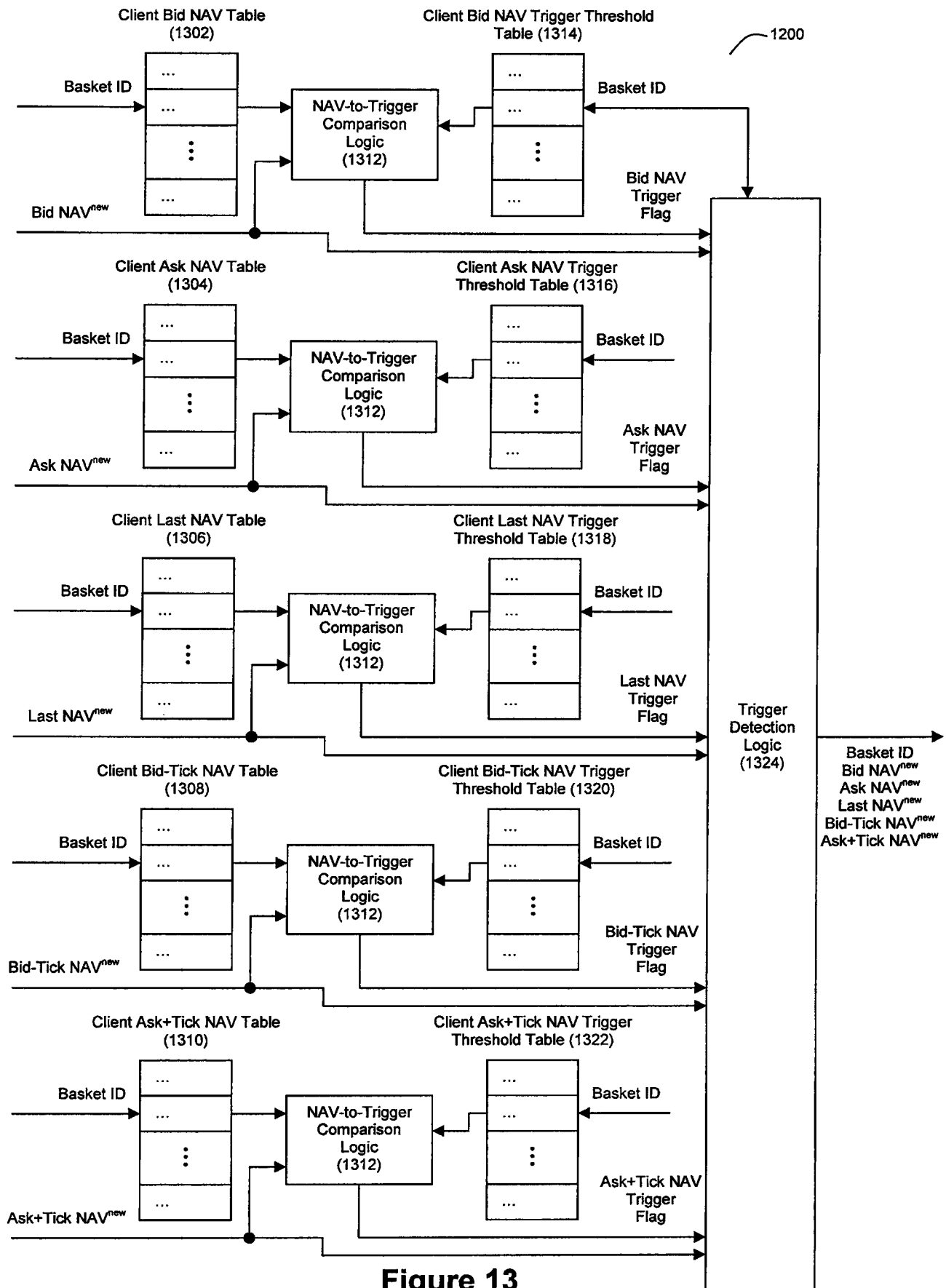
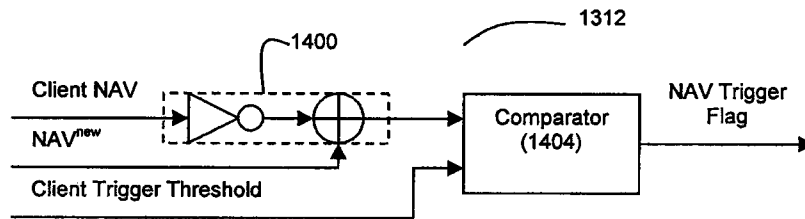
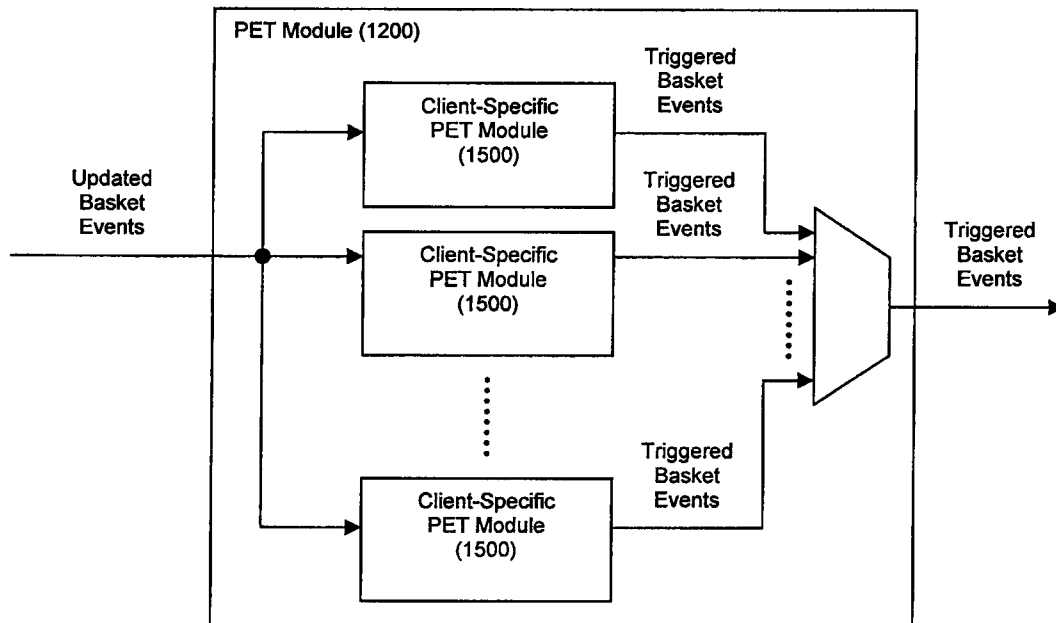


Figure 13

**Figure 14****Figure 15**

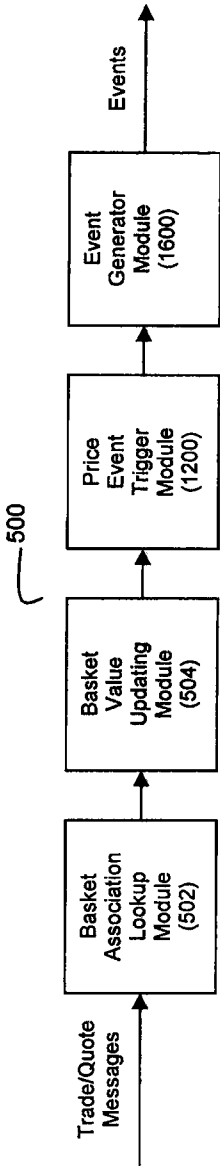


Figure 16

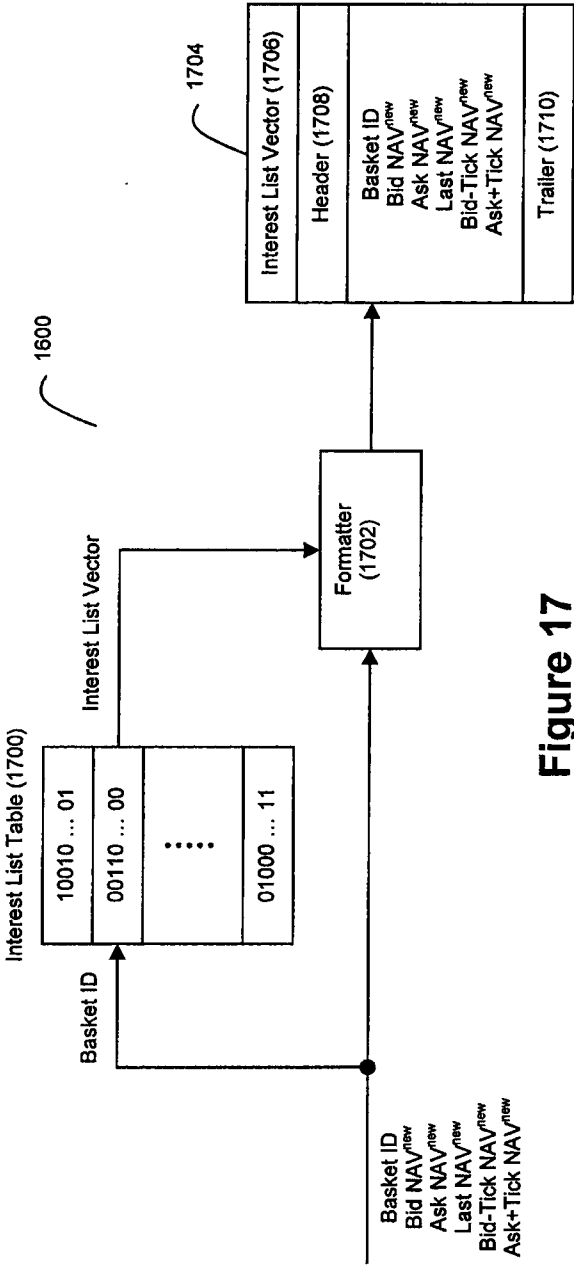


Figure 17

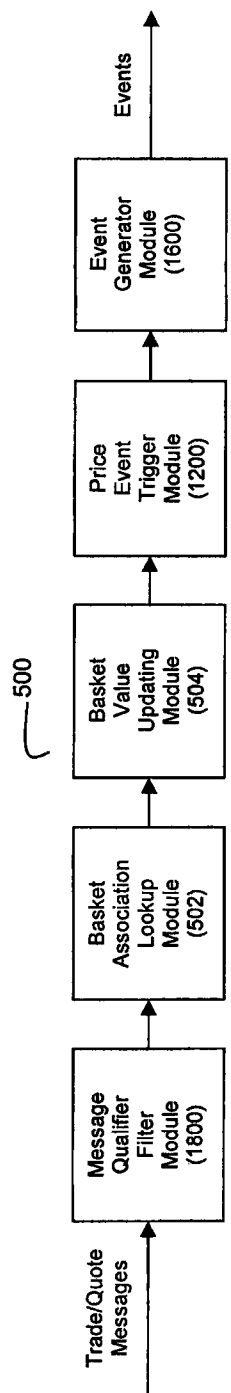


Figure 18

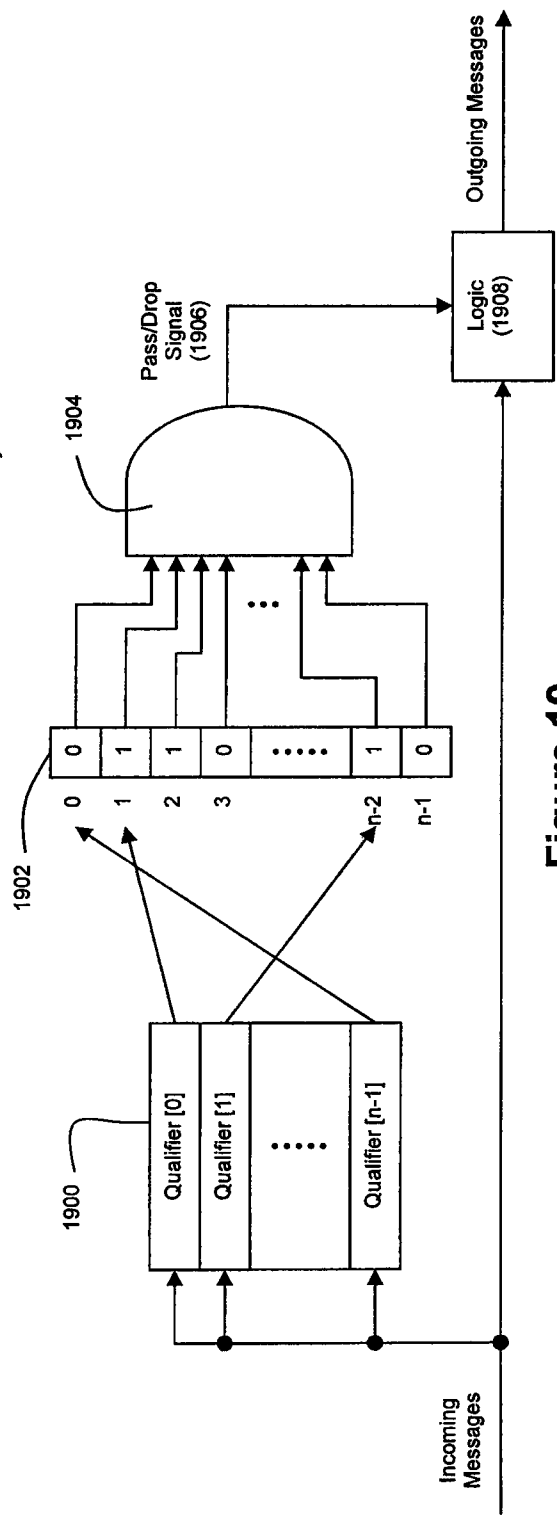


Figure 19

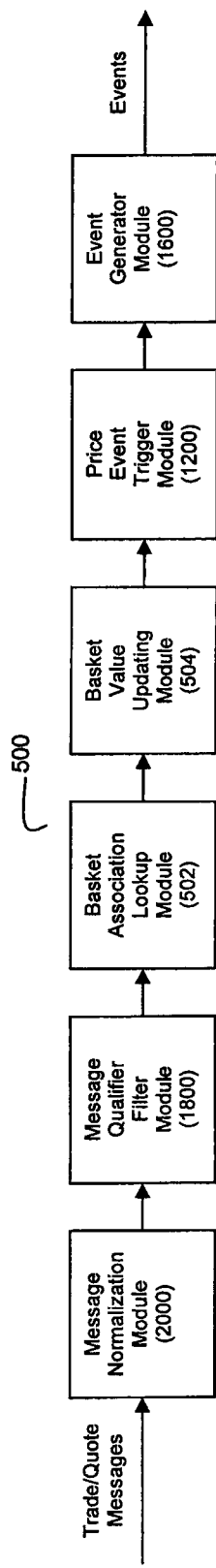


Figure 20

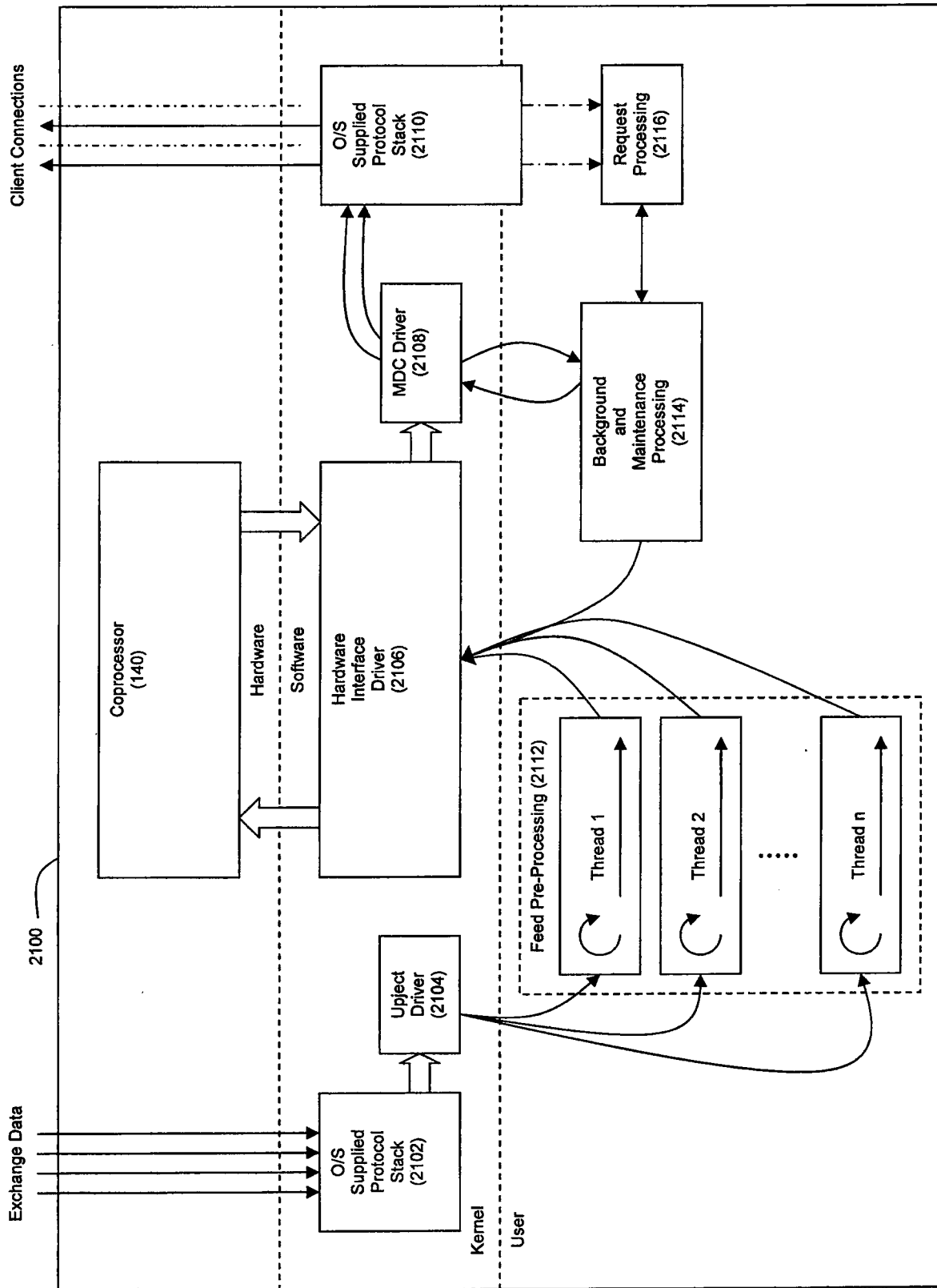


Figure 21