(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2016-95606
(P2016-95606A)
(43)公開日　平成28年5月26日(2016.5.26)

(51)Int.Cl.		F I			テーマコード（参考）
G06F 17/30	(2006.01)	G06F	17/30	415	5B376
G06F 9/445	(2006.01)	G06F	9/06	610A	
		G06F	17/30	110B	

審査請求 未請求　請求項の数 8　ＯＬ　（全 26 頁）

(21)出願番号　特願2014-230387 (P2014-230387)
(22)出願日　平成26年11月13日 (2014.11.13)

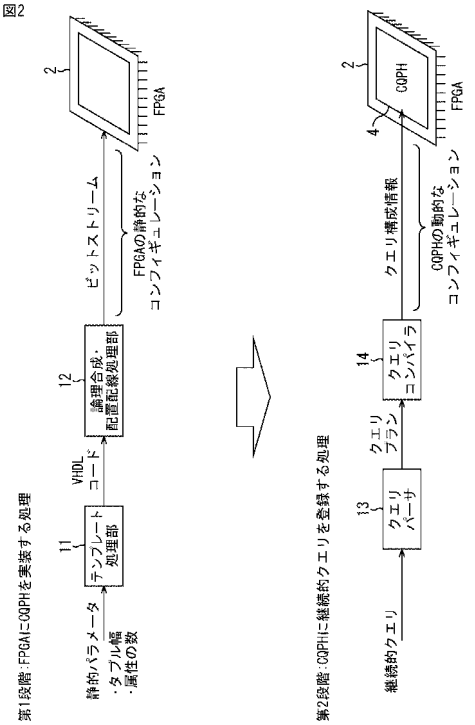
(71)出願人　504133110
国立大学法人電気通信大学
東京都調布市調布ケ丘一丁目５番地１
(74)代理人　100121131
弁理士　西川　孝
(74)代理人　100082131
弁理士　稲本　義雄
(72)発明者　吉永　努
東京都調布市調布ケ丘一丁目５番地１　国立大学法人電気通信大学内
(72)発明者　オゲ　ヤースィン
東京都調布市調布ケ丘一丁目５番地１　国立大学法人電気通信大学内

最終頁に続く

(54)【発明の名称】データ処理装置およびデータ処理方法、並びにプログラム

(57)【要約】
【課題】リアルタイム性が要求されるデータ処理を、クエリの書き換えに対してインタラクティブに対応する。
【解決手段】リアルタイム性が要求されるデータ処理を行うデータ処理部の前段に設けられ、データ処理部におけるデータ処理に対して与えられるクエリ処理を行うクエリ処理部は、動的再構成可能な論理回路として設計されており、データ処理部がデータ処理を行う前段階で、補助処理部の基本的な環境設定を行う第１段階処理が実行され、データ処理部がデータ処理を行っている最中の段階で、補助処理部に対して継続的に与えられる要求に応じて環境設定を変更する第２段階処理とが実行される。本技術は、例えば、データストリーム処理を行うデータ処理装置に適用できる。
【選択図】図２

図2



【特許請求の範囲】
【請求項１】
リアルタイム性が要求されるデータ処理を行うデータ処理部の前段に設けられる再構成可能な論理回路に実装され、前記データ処理部がデータ処理を行う際に必要となる補助的な処理を行う補助処理部と、
前記データ処理部がデータ処理を行う前の段階で、前記補助処理部の基本的な環境設定を行う第１段階処理、および、前記データ処理部がデータ処理を行っている最中の段階で、前記補助処理部に対して継続的に与えられる要求に応じて環境設定を変更する第２段階処理を実行する実装処理部と
を備えるデータ処理装置。
【請求項２】
前記補助処理部は、前記データ処理部におけるデータ処理に対して与えられるクエリに対する処理を行うクエリ処理部である
請求項１に記載のデータ処理装置。
【請求項３】
前記実装処理部は、前記第１段階処理において、静的なパラメータとして与えられたタプル幅および属性の数に基づいて、前記再構成可能な論理回路に前記クエリ処理部を実装する処理を行う
請求項２に記載のデータ処理装置。
【請求項４】
前記実装処理部は、前記第２段階処理において、与えられたクエリを前記クエリ処理部に登録して実行可能な状態とする処理を行う
請求項２または３に記載のデータ処理装置。
【請求項５】
前記クエリ処理部は、複数のコンフィギュラブルなハードウェアモジュールにより構成される
請求項２乃至４のいずれかに記載のデータ処理装置。
【請求項６】
前記クエリ処理部は、データストリームに対するフィルタリング処理、グルーピング処理、ウィンドウ処理、および集約処理のうち、少なくともいずれか１つを含む処理を実行する
請求項２乃至５のいずれかに記載のデータ処理装置。
【請求項７】
リアルタイム性が要求されるデータ処理を行うデータ処理部の前段に設けられる再構成可能な論理回路に実装され、前記データ処理部がデータ処理を行う際に必要となる補助的な処理を行う補助処理部と、
前記再構成可能な論理回路および前記補助処理部に対する処理を実行する実装処理部と
を備えるデータ処理装置のデータ処理方法において、
前記データ処理部がデータ処理を行う前の段階で、前記補助処理部の基本的な環境設定を行う第１段階処理が実行され、
前記データ処理部がデータ処理を行っている最中の段階で、前記補助処理部に対して継続的に与えられる要求に応じて環境設定を変更する第２段階処理が実行される
ステップを含むデータ処理方法。
【請求項８】
リアルタイム性が要求されるデータ処理を行うデータ処理部の前段に設けられる再構成可能な論理回路に実装され、前記データ処理部がデータ処理を行う際に必要となる補助的な処理を行う補助処理部と、
前記再構成可能な論理回路および前記補助処理部に対する処理を実行する実装処理部と
を備えるデータ処理装置のプログラムにおいて、
前記データ処理部がデータ処理を行う前の段階で、前記補助処理部の基本的な環境設

定を行う第１段階処理が実行され、
　前記データ処理部がデータ処理を行っている最中の段階で、前記補助処理部に対して継続的に与えられる要求に応じて環境設定を変更する第２段階処理が実行される
ステップを含む処理をコンピュータに実行させるプログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本開示は、データ処理装置およびデータ処理方法、並びにプログラムに関し、特に、リアルタイム性が要求されるデータ処理の処理中にインタラクティブな対応を行うことができるデータ処理装置およびデータ処理方法、並びにプログラムに関する。
【背景技術】
【０００２】
近年、大規模データ処理の分野において、データストリーム処理技術に対する必要性および注目度が非常に高まってきている。データストリーム処理とは、実世界の活動や業務などにより常時発生する大量の時系列データをリアルタイムに処理・分析する技術のことである。
【０００３】
ここで、本明細書において、データストリーム処理で処理対象となるデータストリームとは、映像・音声ストリームのような論理的に継続する一つのデータではなく、それぞれが論理的に独立した大量の時系列データを示すものである。具体的には、金融アプリケーションにおける株価データ、ITシステムで発生する各種ログデータ、各種センサやRFID（Radio-Frequency Identification）デバイスなどから出力される数値データ、または、走行する車両や人などの位置情報が、データストリームとして例示される。なお、これらの例示に限定されることなく、それぞれが論理的に独立した大量の時系列データをデータストリーム処理の処理対象とすることができる。
【０００４】
データストリーム処理は、当初、金融・証券分野を中心に利用されてきたが、最近では、多種大量な時系列データ（いわゆるビッグデータ）を扱う様々な分野で、リアルタイムな「価値」を効率的に抽出する技術として利用され始めている。そして、今後、大規模化し種類も増大するデータストリームに対して、低コストかつリアルタイムでデータストリーム処理を行う技術の重要度が増加することが想定される。
【０００５】
例えば、非特許文献１では、データストリームに対する解析処理を簡単に扱うためのリアルタイムデータ処理基盤として、データストリーム管理システム（DSMS：Data Stream Management System）のアーキテクチャが開示されている。DSMSでは、非特許文献２に開示されている継続的クエリ（以下、単にクエリと称する）を用いてデータストリーム処理が実現される。一般に、DSMSでは、ユーザが簡単に記述できるSQLライクな宣言的言語でクエリが記述される。
【０００６】
一般的に、データストリームの各データレコード（以下、タプルと呼ぶ）はDSMSに継続的に到着し続けるため、それらのタプルの総量は論理的には無限である。そのためDSMSでは、データストリームの終了を待つことなく、システムに到着し続けるタプルに対して最新10秒間などの時間幅もしくは最新10000件などのタプルの個数を指定し、データストリームの一部を切り出す「ウィンドウ」と呼ばれる概念が導入されている。ウィンドウ指定を含むクエリに関しては、非特許文献３に詳細に開示されている。
【０００７】
また、非特許文献１に開示されているDSMSは、株価の変動や傾向をリアルタイムにモニタリングし、即座に売買を行う金融アプリケーションなど、高レートで発生するデータをリアルタイムに解析する様々なアプリケーションへの適用が期待されている。このようなアプリケーションでは、データストリームにおけるタプルの到着頻度が時に極めて高くな

り、大量のタプルが短時間内にバースト的に到着することがある。

【０００８】

具体例として、非特許文献４が対象としている金融アプリケーションでは、ネットワークから到着する１秒間当たり数１００万パケットのデータに対してマイクロ秒（μ秒）レベルの超高速なシステム応答性能が要求される。また、非特許文献４が対象としている金融応用などのアプリケーションでは、全ての入力タプルを反映した正確な計算結果が求められるため、サンプリングなどによる近似計算は許されない。

【０００９】

このようなアプリケーションに対応するためには、連続的に到着するタプルに対して、取りこぼしなくクエリ処理を実行することが求められる。そのため、タプルの到着頻度が高い場合、連続的に到着するタプルを処理するために非常に高い計算能力が求められる。このような高い計算能力が要求される分野（例えば、金融・通信応用などの分野であるが、それらの分野に必ずしも限定されない）では、非特許文献１のように標準的なプロセッサの利用を前提とした場合、レイテンシ（latency）などの性能要件を満足することができないことがある。

【００１０】

そこで、標準的なプロセッサでは実現困難な性能要求を満たす方法として、専用ハードウェアを活用することが有力な選択肢の１つとして挙げられる。専用ハードウェアでは、内部に搭載可能な多数の演算ユニットの並列動作により高い処理性能を得ることができる。また、近年では開発・製造コストの観点から、専用アーキテクチャをASIC（Application-Specific Integrated Circuit）として独自に生産するのではなく、FPGA（Field-Programmable Gate Array）を用いて実現する手法が積極的に採用されている。

【００１１】

例えば、非特許文献４では、ユーザの所望するクエリをFPGA上に実現する方法が開示されている。具体的には、データストリーム処理を行うための基本要素をハードウェア記述言語（HDL：Hardware Description Language）ライブラリとして用意する。そして、ユーザによって与えられるクエリに応じて、データストリーム処理を行うための基本要素を組み合わせ、クエリ処理専用のハードウェアのHDLコードを自動的に生成する。非特許文献４に開示されている方法は、フィルタリングおよびスライディングウィンドウを用いた集約（aggregation）クエリに対応している。

【００１２】

しかしながら、非特許文献４に開示されている方法には重大な欠点が２つ存在する。１つ目の欠点は、ウィンドウ指定を含む集約クエリにおいて、ウィンドウ幅とスライド幅の比率に応じてハードウェア・リソース使用量が増加してしまう点である。１つ目の欠点については、非特許文献５に開示されている方法をハードウェア設計に応用することによって解決することができる。具体的な解決方法は、非特許文献６に開示されている。

【００１３】

また、非特許文献５に関連する既存技術として特許文献１が挙げられる。しかしながら、特許文献１に開示されている方法は標準的なプロセッサ上で動作するソフトウェアを前提とする点、および、専用ハードウェアとしての実現方法が考慮されない点において、非特許文献６に開示されている方法と異なる。

【００１４】

そして、非特許文献４に開示されている方法の２つ目の欠点は、クエリのコンパイル時間と、論理回路の構成情報をFPGAへダウンロードする際に要する時間との合計が無視できないほど大きくなってしまう点である。即ち、HDLからFPGA上で実行可能な回路情報を構成するには、論理合成と配置配線が必要であるため、ユーザによって与えられたクエリから大規模なHDLが生成された場合には、論理合成と配置配線に要する時間が増加してしまう。その結果、数時間以上のコンパイル時間が必要な場合がある。

【００１５】

従って、非特許文献４に開示されている方法では、頻繁な新規クエリの登録や現在実行

中のクエリの書き換えなどに対応することは非常に困難である。言い換えると、非特許文献４に開示されている方法では、ユーザからの問い合わせ（クエリ）に対してインタラクティブに対応することができない。リアルタイム性が重視されるデータストリーム処理において、リアルタイムにインタラクティブな対応ができない点は大きな問題である。非特許文献６に開示されている方法も同様の問題点を抱えており、この２つ目の欠点を解決する方法は、これまでに実現されていない。また、非特許文献７には、ウィンドウ指定を含む集約クエリに関する技術が開示されている。

【先行技術文献】

【特許文献】

【００１６】

【特許文献１】特許第５３３７４４７号

【非特許文献】

【００１７】

【非特許文献１】Y. Ahmad and U. Cetintemel, "Data stream management architectures and prototypes," in Encyclopedia of Database Systems, L. Liu and M. T. Ozsu, Eds. Springer US, 2009, pp. 639-643.

【非特許文献２】S. Babu, "Continuous query," in Encyclopedia of Database Systems, L. Liu and M. T. Ozsu, Eds. Springer US, 2009, pp. 492-493.

【非特許文献３】W. G. Aref, "Window-based query processing," in Encyclopedia of Database Systems, L. Liu and M. T. Ozsu, Eds. Springer US, 2009, pp. 3533-3538.

【非特許文献４】R. Mueller et al., "Streams on wires: a query compiler for FPGAs," 35th International Conference on Very Large Data Bases (VLDB), vol. 2(1), pp. 229-240, 2009.

【非特許文献５】J. Li et al., "No pane, no gain: efficient evaluation of sliding-window aggregates over data streams," ACM SIGMOD Record, volume 34, issue 1, pp.39-44, 2005.

【非特許文献６】Y. Oge et al., "An Efficient and Scalable Implementation of Sliding-Window Aggregate Operator on FPGA," 1st International Symposium on Computing and Networking (CANDAR'13), pp.112-121, 2013.

【非特許文献７】J. Li et al., "Semantics and evaluation techniques for window aggregates in data streams," 2005 ACM SIGMOD International Conference on Management of Data (SIGMOD), pp.311-322, 2005.

【発明の概要】

【発明が解決しようとする課題】

【００１８】

上述したように、従来の技術においては、クエリの書き換えに対して、数時間以上のコンパイル時間が必要なことがあった。そのため、リアルタイム性が要求されるデータストリーム処理に対して、その処理中にインタラクティブな対応をすることは困難であった。

【００１９】

本開示は、このような状況に鑑みてなされたものであり、リアルタイム性が要求されるデータ処理の処理中にインタラクティブな対応を行うことができるようにするものである。より具体的には、新規クエリの登録や、実行中のクエリの書き換え、実行中のクエリの削除など、与えられたクエリをクエリ処理部で実行可能な状態にするために必要な一連の処理を、ユーザからのリクエストに応じてランタイム時に動的に行うことができる。

【課題を解決するための手段】

【００２０】

本開示の一側面のデータ処理装置は、リアルタイム性が要求されるデータ処理を行うデータ処理部の前段に設けられる再構成可能な論理回路に実装され、データ処理部がデータ処理を行う際に必要となる補助的な処理を行う補助処理部と、データ処理部がデータ処理を行う前の段階で、補助処理部の基本的な環境設定を行う第１段階処理、および、データ

処理部がデータ処理を行っている最中の段階で、補助処理部に対して継続的に与えられる要求に応じて環境設定を変更する第２段階処理を実行する実装処理部とを備える。
【００２１】
本開示の一側面のデータ処理方法またはプログラムは、リアルタイム性が要求されるデータ処理を行うデータ処理部の前段に設けられる再構成可能な論理回路に実装され、データ処理部がデータ処理を行う際に必要となる補助的な処理を行う補助処理部と、再構成可能な論理回路および補助処理部に対する処理を実行する実装処理部とを備えるデータ処理装置のデータ処理方法またはプログラムにおいて、データ処理部がデータ処理を行う前の段階で、補助処理部の基本的な環境設定を行う第１段階処理が実行され、データ処理部がデータ処理を行っている最中の段階で、補助処理部に対して継続的に与えられる要求に応じて環境設定を変更する第２段階処理が実行されるステップを含む。
【００２２】
本開示の一側面においては、データ処理部がデータ処理を行う前の段階で、補助処理部の基本的な環境設定を行う第１段階処理が実行され、データ処理部がデータ処理を行っている最中の段階で、補助処理部に対して継続的に与えられる要求に応じて環境設定を変更する第２段階処理が実行される。
【発明の効果】
【００２３】
本開示の一側面によれば、リアルタイム性が要求されるデータ処理の処理中（いわゆるランタイム時）にインタラクティブな対応を行うことができる。
【図面の簡単な説明】
【００２４】
【図１】本技術を適用したデータストリーム管理システムの一実施の形態の構成例を示すブロック図である。
【図２】第１段階としてFPGAにCQPHを実装する処理と、第２段階としてCQPHに継続的クエリを登録する処理とについて説明する図である。
【図３】コンフィギュラブルなハードウェアモジュールの構成例を示すブロック図である。
【図４】クエリ処理部の構成例を示すブロック図である。
【図５】選択モジュールの構成例を示すブロック図である。
【図６】選択クエリの一例を示す図である。
【図７】選択モジュールの処理例を説明する図である。
【図８】選択モジュールの処理例を説明する図である。
【図９】選択モジュールの処理例を説明する図である。
【図１０】グループ化モジュールの接続インタフェースの構成例を示すブロック図である。
【図１１】スライディング・ウィンドウを伴う集約クエリの一例を示す図である。
【図１２】グループ化モジュールの処理例を説明する図である。
【図１３】グループ化モジュールの処理例を説明する図である。
【図１４】グループ化モジュールの処理例を説明する図である。
【図１５】ウィンドウ集約モジュールの処理例を説明する図である。
【図１６】ウィンドウ集約モジュールの処理例を説明する図である。
【図１７】本技術を適用したコンピュータの一実施の形態の構成例を示すブロック図である。
【発明を実施するための形態】
【００２５】
以下、本技術を適用した具体的な実施の形態について、図面を参照しながら詳細に説明する。
【００２６】
＜データストリーム管理システムの構成例＞

図１は、本技術を適用したデータストリーム管理システムの一実施の形態の構成例を示すブロック図である。

【００２７】

図１に示すように、データストリーム管理システム１は、FPGA（Field-Programmable Gate Array）２、NIC（Network interface controller）３、CQPH（Configurable Query Processing Hardware）４、CPU（Central Processing Unit）５、メモリ６、記憶部７、出力部８、および実装処理部９を備えて構成される。データストリーム管理システム１は、例えば、それぞれが論理的に独立した大量の時系列データであるデータストリームに対して、リアルタイムにデータストリーム処理を行う。

【００２８】

FPGA２は、ユーザが自由にプログラミングすることができる再構成可能な論理回路であり、例えば、幅広い種類の固定型の計算モジュール、または、コンフィギュラブルな計算モジュールを実装することができる。図１に示す構成例では、NIC３およびCQPH４がFPGA２に実装されており、物理ネットワークインタフェースがFPGA２に直接的に接続されている。

【００２９】

NIC３は、インターネットなどのネットワークを介した通信の制御を行い、データストリーム管理システム１において処理対象となるデータストリームを取得して、CQPH４に供給する。

【００３０】

CQPH４は、例えば、リレーショナルデータベースの操作を行うための言語の一つであるSQLベースの問い合わせ（クエリ）をワイヤスピードで実現する動的再構成可能なプログラマブルハードウェアであり、例えば、CPU５がデータ処理を行う際に必要となる補助的な処理としてクエリ処理を行う。例えば、ランタイム時にユーザによって与えられるクエリをコンパイルすることでクエリ構成情報が生成され、更にそのクエリ構成情報をCQPH４内部のクエリ処理専用回路（例えば、図４のクエリ処理部３１）に動的に登録する。つまり、CQPH４内部に、クエリ構成情報に基づいてクエリ処理のための論理回路が構成される。そして、CQPH４は、NIC３から供給されるデータストリームに対してクエリ処理を行い、その処理後のデータストリームをCPU５に供給する。

【００３１】

CPU５は、CQPH４においてクエリ処理が行われたデータストリームに対して、リアルタイムにデータストリーム処理を行う。例えば、CPU５で実行されるデータストリーム処理は、金融・証券など様々な分野の用途に適したアプリケーションとして用意される。

【００３２】

メモリ６は、例えば、RAM（Random Access Memory）であり、CPU５がデータストリーム処理を行う際に必要となる各種のデータを適宜記憶する。

【００３３】

記憶部７は、例えば、ハードディスクなどの記憶装置であり、CPU５によりデータストリーム処理が行われた処理結果を記憶する。

【００３４】

出力部８は、例えば、CPU５によりデータ処理が行われた処理結果を外部に出力し、例えば、図示しない表示装置に処理結果を表示させたり、図示しない通信装置を介して送信したりする。

【００３５】

実装処理部９は、図２を参照して後述するように、FPGA２にCQPH４を実装する処理、および、CQPH４に継続的クエリを登録する処理を行う。例えば、実装処理部９は、図２に示すように、テンプレート処理部（HDL Template of CQPH）１１、論理合成・配置配線処理部（Synthesis and Place & Route）１２、クエリパーサ（Query Parser）１３、およびクエリコンパイラ（Query Compiler）１４を有して構成される。

【００３６】

このように構成されるデータストリーム管理システム１では、ユーザによって与えられるクエリは、FPGA２の内部に実装されるCQPH４上にクエリプランとして登録される。そして、ネットワークから受信されたデータストリームは、NIC３を介して、CQPH４上に構成されたクエリプランに直接的に供給される。なお、図１に示すデータストリーム管理システム１では、ユーザクエリから出力タプルが生成された場合のみ、CPU５に割り込みが発生することになる。例えば、FPGA２は、DMA（Direct Memory Access）を使用して、メモリ６に出力タプルを書き込み、その後、新しいデータの到着についてCPU５に通知することができる。

【００３７】

このように、データストリーム管理システム１では、リアルタイムにデータストリーム処理を実行する基板に、コプロセッサ（副処理装置）としてCQPH４が設けられている。従って、CQPH４がクエリ処理を実行することによってCPU５の負荷を軽減することができるので、CPU５によるデータストリーム処理に要する時間を大幅に短縮することができる。

【００３８】

即ち、通常では、CPU５がクエリ処理を行っているのに対し、データストリーム管理システム１では、CQPH４が、CPU５に代わってクエリ処理を行うクエリオフロードエンジンとして機能する。例えば、データストリーム管理システム１では、連続して流れ続ける高速なデータストリームに対して、CQPH４が、フィルタリングおよびスライディングウィンドウを用いた集約（aggregation）クエリを行う。このように、データストリーム管理システム１では、クエリ処理を行う専用ハードウェアとしてCQPH４を設けることによって、クエリ処理にかかる遅延時間を大幅に低減することができる。その結果、データストリーム管理システム１では、標準的なCPU５では実現困難なリアルタイム処理（低遅延かつ高スループット）を実現することができる。

【００３９】

例えば、上述の非特許文献４が対象としている金融アプリケーションについて、データストリーム管理システム１では、約９０％のデータがCQPH４において削減され、残りの１０％のデータがCPU５において処理される。従って、データストリーム管理システム１では、CPU５の負荷が軽減する結果、データストリーム処理の全体的な性能の向上を図ることができる。

【００４０】

なお、データストリーム管理システム１では、例えば、FPGA２がCQPH４だけを実装するような構成を採用することができ、CPU５が、CQPH４の内部レジスタ（スレーブレジスタ）にデータストリームを直接的に書き込むようにしてもよい。または、CPU５が、CQPH４と共用するメモリ領域にデータストリームを予め用意してもよい。この構成では、CQPH４は、一般的なコプロセッサとして用いられ、CPU５が、CQPH４に作業要求を送信して処理が実行される。

【００４１】

次に、図２を参照して、FPGA２にCQPH４を実装する処理と、FPGA２に継続的クエリを登録する処理とについて説明する。

【００４２】

データストリーム管理システム１では、ユーザによって与えられるクエリに対してCQPH４がインタラクティブな対応を実現するために、２段階のコンフィギュレーション・アプローチが採用される。即ち、第１段階では、図２の上側に示すように、FPGA２の静的なコンフィギュレーション（基本的な環境設定を行う処理）が行われ、第２段階では、図２の下側に示すように、FPGA２内部に実装されたCQPH４の動的なコンフィギュレーション（継続的に与えられる要求に応じて環境設定を変更する処理）が行われる。

【００４３】

例えば、CQPH４は、高度にパラメータ化されたHDLモデルとして設計されているため、特定のアプリケーションに最適化されたCQPHインスタンスを生成するためには、静的なコンフィギュレーション・パラメータが必要である。一般的なユーザは、最低限必要なコン

フィグレーション・パラメータとして、タプルのサイズ（タプル幅）や属性の数のようなアプリケーションに関連する基本的な情報を提供することができる。一方、ハードウェア設計のスキルを有するユーザは、CQPHテンプレートの設計に含まれるコンフィギュラブルなハードウェアモジュールの数を調整することにより、短時間に様々なアーキテクチャを試しながら、電力や性能、面積などのトレードオフを行うことができる。

【００４４】

そして、CQPH４の実装は、通常のFPGA設計フローに従って行われる。

【００４５】

まず、第１段階において、ユーザによって、タプル幅や属性の数などの静的なパラメータがテンプレート処理部１１に供給される。テンプレート処理部１１は、それらの静的なパラメータに基づいて論理回路の構成情報（VHDLコード）を生成し、その論理回路の構成情報を論理合成・配置配線処理部１２に供給する。

【００４６】

論理合成・配置配線処理部１２は、標準的なFPGAツールチェーンであり、テンプレート処理部１１において生成された論理回路の構成情報をFPGA２へダウンロードすることによって、FPGA２にCQPH４を実装する。

【００４７】

なお、第１段階において行われる一連の処理は、ユーザがランタイムにクエリを実行する前に、一度だけ実施されるものである。そして、静的なコンフィギュレーション・プロセスを経てCQPH４が実装された後、第２段階が行われる。

【００４８】

次に、第２段階において、ユーザによって、継続的クエリがクエリパーサ１３に供給される。クエリパーサ１３は、それらのクエリをパースし、クエリプランをクエリコンパイラ１４に供給する。

【００４９】

クエリコンパイラ１４は、CQPH専用のクエリ構成情報を生成する専用コンパイラであり、クエリパーサ１３から供給されるクエリプランに基づいてクエリをコンパイルする。クエリコンパイラ１４によって生成されたクエリ構成情報は、コンフィギュレーションタプルと呼ばれる単位に分割された後、一連のコンフィギュレーションタプルが、CQPH４にストリーム的に転送される。

【００５０】

このように、データストリーム管理システム１では、CPU５がデータストリーム処理を行う前の段階でFPGA２にCQPH４を実装する第１段階の実装処理を実行するステップ（このステップでハードウェアの実装が完了）と、CPU５がデータストリーム処理を行っている最中の段階（ランタイム時）でFPGA２内部に実装されたCQPH４にクエリ構成情報を登録する第２段階の実装処理（コンフィギュラブルなハードウェアモジュール内部のコンフィギュレーション・レジスタの書き換えが行われる）を実行するステップとによってCQPH４にクエリが登録される。このように、CQPH４が、ランタイム時の動的なクエリの書き換えに柔軟に対応することで、ユーザによって与えられるクエリに対してインタラクティブに対応することができる。

【００５１】

ところで、上述した非特許文献４に開示されているFPGAを用いたアプローチの欠点は、ランタイム時にユーザによって与えられたクエリをコンパイルする際に、クエリの規模に応じて論理合成や配置配線などに要する時間が大幅に増加してしまう点である。これに対し、クエリコンパイラ１４は、クエリをコンパイルする際に論理合成や配置配線などの処理を必要としないため、CQPH４におけるクエリのコンパイル・プロセスは、非特許文献４に開示されている方法の制約を受けることはない。その結果、CQPH４は、ランタイム時にユーザによって与えられるクエリに対して、インタラクティブに対応するために不可欠な要素である高いフレキシビリティを提供することができる。

【００５２】

また、CQPH４内部のクエリ処理専用回路のテンプレート・デザインは、数種類のコンフィギュラブルなハードウェアモジュールによって構成することができる。

【００５３】

図３には、CQPH４に構成されるコンフィギュラブルなハードウェアモジュールの構成例を示すブロック図と、その接続インタフェースが示されている。なお、図３に示す構成例は例示的なものであり、CQPH４に構成されるコンフィギュラブルなハードウェアモジュールは、図示される構成例に限定されるものではない。

【００５４】

図３に示すように、コンフィギュラブルなハードウェアモジュール２１は、接続インタフェースとして、ビットフラグフィールド接続インタフェース２２およびデータフィールド接続インタフェース２３を有している。また、コンフィギュラブルなハードウェアモジュール２１は、コンフィギュレーションデコーダ２４、コンフィギュレーションレジスタ２５、および、プログラマブルハードウェアブロック２６を備えて構成される。

【００５５】

ビットフラグフィールド接続インタフェース２２には、複数のフラグが格納されており、その一つはコンフィギュレーションフラグと称されるものである。ビットフラグフィールド接続インタフェース２２に格納されるフラグ数およびデータフィールド幅は、図２を参照して上述した第１段階の静的なコンフィギュレーションを行う際に与えられるパラメータによって決定される。

【００５６】

データフィールド接続インタフェース２３は、ｎビット幅のデータを表す複数の配線としてみなされる。

【００５７】

また、CQPH４は、複数の種類からなる多数のハードウェアモジュール２１によって構成され、ハードウェアモジュール２１それぞれは、統一された接続インタフェースによって別のハードウェアモジュール２１と接続される。例えば、直前のハードウェアモジュール２１のコンフィギュレーションフラグがアサートされた場合、当該のハードウェアモジュール２１のデータフィールドはクエリ構成情報の一部（すなわち、コンフィギュレーションタプル）としてみなされる。

【００５８】

各コンフィギュレーションタプルは、コンフィギュレーションデータフィールドおよびターゲットIDフィールドからなる２つの主要部分から構成されている。その名前が示すように、コンフィギュレーションデータフィールドは、特定のハードウェアモジュール２１のクエリ構成情報を含む。ターゲットIDフィールドには、各ハードウェアモジュール２１にあらかじめ割り当てられた固有の識別子（ID番号）が含まれている。

【００５９】

コンフィギュレーションデコーダ２４は、コンフィギュレーションタプルが与えられると、そのコンフィギュレーションタプルのターゲットIDフィールドが、コンフィギュレーションデコーダ２４自身のID番号と一致するかどうかを評価する。

【００６０】

例えば、コンフィギュレーションデコーダ２４は、与えられたコンフィギュレーションタプルのターゲットIDフィールドがコンフィギュレーションデコーダ２４自身のID番号と一致する場合、そのコンフィギュレーションタプルに含まれるクエリ構成情報を取り出す。そして、コンフィギュレーションデコーダ２４は、コンフィギュレーションタプルから取り出したクエリ構成情報をコンフィギュレーションレジスタ２５に格納する。

【００６１】

一方、コンフィギュレーションデコーダ２４は、与えられたコンフィギュレーションタプルのターゲットIDフィールドがコンフィギュレーションデコーダ２４自身のID番号と一致しない場合、そのコンフィギュレーションタプルを単に次のハードウェアモジュール２１に転送する。ここで、各ハードウェアモジュール２１は、ランタイム時に単一のコンフ

ィギュレーションタプルによって、わずか１クロックサイクルで動的にコンフィギュレーション可能である。

【００６２】

プログラマブルハードウェアブロック２６は、コンフィギュレーションレジスタ２５に格納されているクエリ構成情報を読み出して、そのクエリ構成情報に従った動作を行う。

【００６３】

このように、ユーザはランタイム時に各ハードウェアモジュール２１のコンフィギュレーションレジスタ２５を動的に書き換えることができる。これにより、ユーザは、インタラクティブにハードウェアモジュール２１内のプログラマブルハードウェアブロック２６の動作を変更することができる。従って、ハードウェアモジュール２１では、新しいクエリを追加したり、既存のクエリを変更または削除したりする際に要する時間を、上述の非特許文献４に開示されている方法と比較して無視できる程度まで短縮することができる。

【００６４】

このようにコンフィギュラブルなハードウェアモジュール２１は構成され、複数のコンフィギュラブルなハードウェアモジュール２１によって、CQPH４のアーキテクチャが構成される。例えば、図１のデータストリーム管理システム１では、フィルタリング、グルーピング、およびウィンドウ指定を含む集約クエリに特化するように構成されたCQPH４のアーキテクチャが採用され、このようなCQPH４のアーキテクチャを、以下適宜、クエリ処理部と称する。

【００６５】

＜クエリ処理部の構成例＞

次に、図４は、CQPH４のアーキテクチャにより実現されるクエリ処理部の構成例を示すブロック図である。

【００６６】

図４に示すように、クエリ処理部３１は、選択モジュール３２、Ｎ個のグループ化モジュール３３－１乃至３３－Ｎ、Ｎ個のウィンドウ集約モジュール３４－１乃至３４－Ｎ、および統合モジュール３５を備えて構成される。また、選択モジュール３２、グループ化モジュール３３－１乃至３３－Ｎ、およびウィンドウ集約モジュール３４－１乃至３４－Ｎは、それぞれコンフィギュラブルなハードウェアモジュール２１（図３）として実装され、以下適宜、それらをハードウェアモジュール２１とも称する。

【００６７】

なお、図４において、各ハードウェアモジュール２１を接続する矢印は、それぞれのハードウェアモジュール２１間におけるデータフローの方向を表している。また、CQPH４のアーキテクチャでは、クエリ処理部３１を構成する各ハードウェアモジュール２１はプッシュベースの処理モデルを採用しており、パイプライン的な動作を行うことができる。さらに、図４に示す各ハードウェアモジュール２１は、１クロックサイクル毎に新しいタプルを受け入れることができ、例えば、ハードウェアモジュール２１の内部における処理は、１クロックサイクルまたは数クロックサイクルで完了する。このように、クエリ処理部３１は、複数のハードウェアモジュール２１により構成されており、粗粒度および細粒度の並列性を有する計算処理を、並列に動作させることができる。

【００６８】

選択モジュール３２は、入力タプルに対してフィルタリング処理を行った後、フィルタをパスしたタプルをグループ化モジュール３３－１に供給する。なお、選択モジュール３２の構成およびフィルタリング処理については、図５乃至図９を参照して後述する。

【００６９】

グループ化モジュール３３－１乃至３３－Ｎは、選択モジュール３２においてフィルタリングされた入力タプルに対してグルーピング処理を行う。例えば、グループ化モジュール３３－１は、グループ化モジュール３３－１に設定されたグループの入力タプルをウィンドウ集約モジュール３４－１に供給し、そのグループ以外の入力タプルをグループ化モジュール３３－２に供給する。グループ化モジュール３３－２は、グループ化モジュール

３３－１と同様に、グループ化モジュール３３－２に設定されたグループの入力タプルをウィンドウ集約モジュール３４－２に供給し、そのグループ以外の入力タプルをグループ化モジュール３３－３に供給する。以下同様の処理を、グループ化モジュール３３－３からグループ化モジュール３３－Ｎまで行うことにより、入力タプルがグループ化される。なお、グループ化モジュール３３－１乃至３３－Ｎによるグルーピング処理については、図１０乃至図１４を参照して後述する。

【００７０】

ウィンドウ集約モジュール３４－１乃至３４－Ｎは、グループ化モジュール３３－１乃至３３－Ｎによってグループ化された入力タプルごとに、入力タプルに対するウィンドウ処理および集約処理を行い、出力タプルを統合モジュール３５に供給する。

【００７１】

図示するように、ウィンドウ集約モジュール３４－１乃至３４－Ｎは、ペインレベルサブクエリ処理部４１－１乃至４１－Ｎ、ペインバッファ４２－１乃至４２－Ｎ、およびウィンドウレベルサブクエリ処理部４３－１乃至４３－Ｎを、それぞれ備えて構成される。即ち、ウィンドウ集約モジュール３４－１は、ペインレベルサブクエリ処理部４１－１、ペインバッファ４２－１、およびウィンドウレベルサブクエリ処理部４３－１が直列に接続されて構成される。また、ウィンドウ集約モジュール３４－２は、ペインレベルサブクエリ処理部４１－２、ペインバッファ４２－２、およびウィンドウレベルサブクエリ処理部４３－２が直列に接続されて構成される。以下同様に、ウィンドウ集約モジュール３４－３乃至３４－Ｎが構成される。なお、ウィンドウ集約モジュール３４－１乃至３４－Ｎによるウィンドウ処理および集約処理については、図１５および図１６を参照して後述する。

【００７２】

統合モジュール３５は、Ｎ個のFIFOバッファ４４－１乃至４４－Ｎを備えて構成され、ウィンドウ集約モジュール３４－１乃至３４－Ｎから供給されるＮ本の出力ストリームをバッファリングし、単一のストリームとして出力する。なお、統合モジュール３５は、ランタイム時のコンフィギュラビリティを必要としないため、固定のハードウェアブロックとして実装される。具体的には、非特許文献４に開示されている方法を用いて統合モジュール３５を実装することができる。

【００７３】

＜選択モジュールの構成例＞

次に、図５は、図４の選択モジュール３２の構成例を示すブロック図である。

【００７４】

図５には、選択判断（Selection Predicate）を行うモジュールの数が４に指定され、２分木構造の回路（Boolean Expression Tree）の数が３に指定された場合の選択モジュール３２が示されている。

【００７５】

図５に示すように、選択モジュール３２は、４つの選択判断モジュール６１－１乃至６１－４、４つのレジスタ６２－１乃至６２－４、９つのバイナリレデューサモジュール６３－１乃至６３－９、および、３つのレジスタ６４－１乃至６４－３を備えて構成される。また、選択判断モジュール６１－１乃至６１－４およびバイナリレデューサモジュール６３－１乃至６３－９は、それぞれコンフィギュラブルなハードウェアモジュール２１（図３）として実装される。

【００７６】

図示するように、選択モジュール３２では、複数のバイナリレデューサモジュール６３を用いて２分木構造の回路が構成され、２分木構造の回路の最下位のノードの入力ポートにレジスタ６２が接続される。図５の構成例では、バイナリレデューサモジュール６３－１乃至６３－３により第１の２分木構造の回路が構成され、バイナリレデューサモジュール６３－４乃至６３－６により第２の２分木構造の回路が構成され、バイナリレデューサモジュール６３－７乃至６３－９により第３の２分木構造の回路が構成されている。また

、選択モジュール３２は、選択判断モジュール６１－１乃至６１－４の出力が、レジスタ６２－１乃至６２－４それぞれを通して、複数の２分木構造の回路によって共有されるように配線が設計されている。

【００７７】

なお、複数のバイナリレデューサモジュール６３により構成される２分木構造の回路のサイズは、選択判断モジュール６１の数によって自動的に決定される。例えば、ユーザは、FPGA２の静的なコンフィギュレーション（図２の第１段階）を行う際に、選択判断モジュール６１の数、および、複数のバイナリレデューサモジュール６３により構成される２分木構造の回路の数を静的なパラメータとして自由に指定することができる。

【００７８】

また、選択モジュール３２では、選択判断モジュール６１－１乃至６１－４による処理結果がレジスタ６２－１乃至６１－４に保持される第１ステージと、バイナリレデューサモジュール６３－１乃至６３－９による処理結果がレジスタ６４－１乃至６４－３に保持される第２ステージとで処理が行われる。

【００７９】

選択判断モジュール６１－１乃至６１－４の内部には、フィルタ条件を設定するための表式（SQLにおけるWHERE句）で指定されている条件判定を行うロジックが実装される。

【００８０】

レジスタ６２－１乃至６２－４は、それぞれ選択判断モジュール６１－１乃至６１－４から出力される１ビットのフラグを保持し、出力する。

【００８１】

バイナリレデューサモジュール６３－１乃至６３－９の内部には、２入力１出力のロジックが実装される。

【００８２】

レジスタ６４－１は、バイナリレデューサモジュール６３－３から出力される１ビットのフラグを保持し、出力する。同様に、レジスタ６４－２および６４－３は、バイナリレデューサモジュール６３－６および６３－９それぞれから出力される１ビットのフラグを保持し、出力する。

【００８３】

このように構成される選択モジュール３２において、図６に示すような３つのクエリが選択モジュール３２に順次登録された場合におけるフィルタリング処理について、図７乃至図９を参照して説明する。

【００８４】

図６には、３つの属性から構成される入力ストリームＳ：＜Ａ、Ｂ、Ｃ＞に対する単純な選択クエリの例が示されており、SQLライクなクエリではWHERE句を用いてフィルタの条件が指定されている。

【００８５】

例えば、図６に示すクエリＱ１を選択モジュール３２に登録する場合、図７に示すように、選択判断モジュール６１－１には、クエリＱ１により指定されるフィルタ条件（Ａ＝１）が設定され、バイナリレデューサモジュール６３－１および６３－３にロジック（LEFT）が設定される。これにより、バイナリレデューサモジュール６３－３からフラグ０が出力される構成となる。このとき、フラグ０がクエリＱ１に対応する形となっている。

【００８６】

また、図６に示すクエリＱ１に続いて、図６に示すクエリＱ２を登録する場合、図８に示すように、選択判断モジュール６１－１の出力は、レジスタ６２－１を通して、クエリＱ１と共有される。従って、クエリＱ１により既に指定されたフィルタ条件（A=1）を再設定する必要は無い。また、選択判断モジュール６１－２には、クエリＱ２により指定されるフィルタの条件（Ｂ＞２）が設定され、バイナリレデューサモジュール６３－４にロジック（AND）が設定され、バイナリレデューサモジュール６３－６にロジック（LEFT）が設定される。これにより、バイナリレデューサモジュール６３－３からフラグ０が出力

されるとともに、バイナリレデューサモジュール６３－６からフラグ１が出力される構成となる。このとき、フラグ０がクエリＱ１に対応し、フラグ１がクエリＱ２に対応する形となっている。

【００８７】

また、図６に示すクエリＱ２に続いて、図６に示すクエリＱ３を登録する場合、図９に示すように、選択判断モジュール６１－１の出力は、レジスタ６２－１を通してクエリＱ１およびＱ２と共有される。また、同様に、選択判断モジュール６１－２の出力は、レジスタ６２－２を通じてクエリＱ１およびクエリＱ２と共有される。

【００８８】

さらに、選択判断モジュール６１－３には、クエリＱ３により指定されるフィルタ条件（Ｃ＜３）が設定され、バイナリレデューサモジュール６３－７にロジック（OR）が設定され、バイナリレデューサモジュール６３－８にロジック（LEFT）が設定され、バイナリレデューサモジュール６３－９にロジック（AND）が設定される。これにより、バイナリレデューサモジュール６３－３からフラグ０が出力され、バイナリレデューサモジュール６３－６からフラグ１が出力されるとともに、バイナリレデューサモジュール６３－９からフラグ２が出力される構成となる。このとき、フラグ０がクエリＱ１に対応し、フラグ１がクエリＱ２に対応し、フラグ２がクエリＱ３に対応する形となっている。

【００８９】

以上のように、選択モジュール３２では、コンフィギュラブルなハードウェアモジュール２１により実装される選択判断モジュール６１およびバイナリレデューサモジュール６３によって、入力タプルに対するフィルタリング処理を行うことができる。

【００９０】

＜グループ化モジュールの構成例＞

次に、図１０は、図４のグループ化モジュール３３の接続インタフェースの構成例を示すブロック図である。

【００９１】

グループ化モジュール３３は、選択モジュール３２によるフィルタリングをパスした入力タプルに対してグルーピング処理を行う。

【００９２】

図１０には、直列的に接続される複数のグループ化モジュール３３のうちの、ｎ番目のグループ化モジュール３３［ｎ］、グループ化モジュール３３［ｎ］の前段に配置されているグループ化モジュール３３［ｎ－１］、および、グループ化モジュール３３［ｎ］の後段に配置されているグループ化モジュール３３［ｎ＋１］が示されている。

【００９３】

例えば、ユーザは、FPGA２の静的なコンフィギュレーション（図２の第１段階）を行う際に、グループ化モジュール３３のモジュール数を静的なパラメータとして自由に設定することができる。クエリ処理部３１では、各グループ化モジュール３３を、１入力２出力のノードとしてみなすことができる。

【００９４】

図１０に示すように、グループ化モジュール３３は、接続インタフェースとして、入力ポート７１および７２、第１出力ポート７３および７４、並びに、第２出力ポート７５および７６を備える。また、入力ポート７１、第１出力ポート７３、および第２出力ポート７５は、ビットフラグ領域（bit flag field）であり、入力ポート７２、第１出力ポート７４、および第２出力ポート７６は、データ領域（data field）である。

【００９５】

図示するように、グループ化モジュール３３［ｎ］の入力ポート７１および７２は、グループ化モジュール３３［ｎ－１］の第１出力ポート７３および７４にそれぞれ接続される。また、グループ化モジュール３３［ｎ］の第１出力ポート７３および７４は、グループ化モジュール３３［ｎ＋１］の入力ポート７１および７２にそれぞれ接続される。また、グループ化モジュール３３［ｎ］の第２出力ポート７５および７６は、ウィンドウ集約

モジュール３４（図４）に接続される。

【００９６】

なお、クエリ処理部３１では、グループ化モジュール３３－１乃至３３－Ｎによるグルーピング処理を一種のルーティング問題として見なすため、ルーティング・ロジックを実装する必要がある。また、SQLライクなクエリでは、どの属性を対象にグループ化するのかを指定するための表式（GROUP-BY句）を用いてグルーピングする属性を指定することができる。

【００９７】

図１１には、3つの属性から構成される入力ストリームbids：＜item-id, bid-price, timestamp＞に対するGROUP-BY句を含む集約クエリの例が示されている。クエリQ4では、item-id属性ごとにグルーピング処理を行っており、timestamp属性を用いてウィンドウ処理が行われている。図１１に示す集約クエリでは、ウィンドウ幅（RANGE）が４分として指定され、スライド幅（SLIDE）が１分として指定されている。

【００９８】

なお、ウィンドウ指定を含む集約クエリに関しては、上述の非特許文献７に詳細に開示されており、図１１のクエリＱ４のウィンドウは、非特許文献７に開示されている方法に従って定義されている。

【００９９】

図１２乃至図１４を参照して、４つのグループ化モジュール３３－１乃至３３－４によるグルーピング処理の例について説明する。

【０１００】

まず、初期状態として、グループ化モジュール３３－１乃至３３－４いずれも、空レジスタには、「１」が設定されており、グループ・レジスタには、グループが設定されていないことを示す「None」が設定されている。

【０１０１】

そして、第１ステップにおいて、グループ化モジュール３３－１にグループGR00の入力タプルが供給されると、グループ化モジュール３３－１の空レジスタには「０」が設定されるとともに、グループ化モジュール３３－１のグループ・レジスタには「GR00」が設定される。

【０１０２】

次に、第２ステップにおいて、グループ化モジュール３３－１は、グループGR00の入力タプルをウィンドウ集約モジュール３４－１（図４）に供給する。また、このとき、次の入力タプルとして、グループGR02の入力タプルがグループ化モジュール３３－１に供給される。

【０１０３】

第３ステップにおいて、グループ化モジュール３３－１は、自身に「GR00」が設定されているため、グループGR02の入力タプルをグループ化モジュール３３－２に供給する。これにより、グループ化モジュール３３－２の空レジスタには「０」が設定されるとともに、グループ化モジュール３３－２のグループ・レジスタには「GR02」が設定される。また、このとき、次の入力タプルとして、グループGR00の入力タプルがグループ化モジュール３３－１に供給される。

【０１０４】

第４ステップにおいて、グループ化モジュール３３－１は、自身に設定されているグループGR00の入力タプルをウィンドウ集約モジュール３４－１に供給する。同様に、グループ化モジュール３３－２は、自身に設定されているグループGR02の入力タプルをウィンドウ集約モジュール３４－２に供給する。また、このとき、次の入力タプルとして、グループGR01の入力タプルがグループ化モジュール３３－１に供給される。

【０１０５】

次に、図１３に示すように、第５ステップにおいて、グループ化モジュール３３－１は、自身に「GR00」が設定されているため、グループGR01の入力タプルをグループ化モジュ

ール３３－２に供給する。また、このとき、次の入力タプルとして、グループGR02の入力タプルがグループ化モジュール３３－１に供給される。

【０１０６】

第６ステップにおいて、グループ化モジュール３３－２は、自身に「GR02」が設定されているため、グループGR01の入力タプルをグループ化モジュール３３－３に供給する。これにより、グループ化モジュール３３－３の空レジスタには「０」が設定されるとともに、グループ化モジュール３３－３のグループ・レジスタには「GR01」が設定される。グループ化モジュール３３－１は、グループGR02の入力タプルをグループ化モジュール３３－２に供給する。また、このとき、次の入力タプルとして、グループGR03の入力タプルがグループ化モジュール３３－１に供給される。

【０１０７】

第７ステップにおいて、グループ化モジュール３３－２は、自身に設定されているグループGR02の入力タプルをウィンドウ集約モジュール３４－２に供給する。同様に、グループ化モジュール３３－３は、自身に設定されているグループGR01の入力タプルをウィンドウ集約モジュール３４－３に供給する。グループ化モジュール３３－１は、グループGR03の入力タプルをグループ化モジュール３３－２に供給する。また、このとき、次の入力タプルとして、グループGR01の入力タプルがグループ化モジュール３３－１に供給される。

【０１０８】

第８ステップにおいて、グループ化モジュール３３－２は、グループGR03の入力タプルをグループ化モジュール３３－３に供給し、グループ化モジュール３３－１は、グループGR01の入力タプルをグループ化モジュール３３－２に供給する。また、このとき、次の入力タプルとして、グループGR00の入力タプルがグループ化モジュール３３－１に供給される。

【０１０９】

第９ステップにおいて、グループ化モジュール３３－３は、自身に「GR01」が設定されているため、グループGR03の入力タプルをグループ化モジュール３３－４に供給する。これにより、グループ化モジュール３３－４の空レジスタには「０」が設定されるとともに、グループ化モジュール３３－４のグループ・レジスタには「GR03」が設定される。グループ化モジュール３３－２は、グループGR01の入力タプルをグループ化モジュール３３－３に供給し、グループ化モジュール３３－１は、グループGR00の入力タプルをウィンドウ集約モジュール３４－１に供給する。また、このとき、次の入力タプルとして、グループGR02の入力タプルがグループ化モジュール３３－１に供給される。

【０１１０】

このように、第９ステップまでに、グループ化モジュール３３－１にはグループGR00が設定され、グループ化モジュール３３－２にはグループGR02が設定され、グループ化モジュール３３－３にはグループGR01が設定され、グループ化モジュール３３－４にはグループGR03が設定される。

【０１１１】

次に、図１４に示すように、第１０ステップにおいて、グループ化モジュール３３－４は、自身に設定されているグループGR03の入力タプルをウィンドウ集約モジュール３４－４に供給する。同様に、グループ化モジュール３３－３は、自身に設定されているグループGR01の入力タプルをウィンドウ集約モジュール３４－３に供給する。グループ化モジュール３３－１は、グループGR02の入力タプルをグループ化モジュール３３－２に供給する。また、このとき、次の入力タプルとして、グループGR03の入力タプルがグループ化モジュール３３－１に供給される。

【０１１２】

第１１ステップにおいて、グループ化モジュール３３－２は、自身に設定されているグループGR02の入力タプルをウィンドウ集約モジュール３４－２に供給する。グループ化モジュール３３－１は、グループGR03の入力タプルをグループ化モジュール３３－２に供給する。また、このとき、次の入力タプルとして、グループGR01の入力タプルがグループ化

モジュール３３－１に供給される。
【０１１３】
第１２ステップにおいて、グループ化モジュール３３－２は、グループGR03の入力タプルをグループ化モジュール３３－３に供給し、グループ化モジュール３３－１は、グループGR01の入力タプルをグループ化モジュール３３－２に供給する。また、このとき、次の入力タプルとして、グループGR02の入力タプルがグループ化モジュール３３－１に供給される。
【０１１４】
第１３ステップにおいて、グループ化モジュール３３－３は、グループGR03の入力タプルをグループ化モジュール３３－４に供給し、グループ化モジュール３３－２は、グループGR01の入力タプルをグループ化モジュール３３－３に供給する。また、グループ化モジュール３３－１は、グループGR02の入力タプルをグループ化モジュール３３－２に供給する。また、このとき、次の入力タプルとして、グループGR03の入力タプルがグループ化モジュール３３－１に供給される。
【０１１５】
第１４ステップにおいて、グループ化モジュール３３－４は、グループGR03の入力タプルをウィンドウ集約モジュール３４－４に供給し、グループ化モジュール３３－３は、グループGR01の入力タプルをウィンドウ集約モジュール３４－３に供給する。同様に、グループ化モジュール３３－２は、グループGR02の入力タプルをウィンドウ集約モジュール３４－２に供給する。また、グループ化モジュール３３－１は、グループGR03の入力タプルをグループ化モジュール３３－２に供給する。また、このとき、次の入力タプルとして、グループGR00の入力タプルがグループ化モジュール３３－１に供給される。
【０１１６】
以上のように、グループ化モジュール３３は、自身にグループが設定されていないときに供給される入力タプルのグループを設定する。そして、グループ化モジュール３３は、自身に設定されたグループの入力タプルをウィンドウ集約モジュール３４に供給し、自身に設定されていないグループの入力タプルを次のグループ化モジュール３３に供給する。これにより、複数のグループ化モジュール３３によるグルーピング処理が行われる。
【０１１７】
＜ウィンドウ集約モジュール３４における処理例＞
次に、図１５および図１６を参照して、図４のウィンドウ集約モジュール３４の処理例について説明する。
【０１１８】
例えば、図１５に示す例では、左側から右側に向かって経過する時刻timeが示されており、ウィンドウ幅（Range）が３分として指定され、スライド幅（Slide）が１分として指定されている。これに応じ、ペインレベルサブクエリ処理部４１は、グループ化モジュール３３から供給されるタプルを、「ペイン」と呼ばれる１分ごとの間隔（サブ・ウィンドウ）に区切って集約計算を実行する。また、ペインレベルサブクエリ処理部４１は、前述の集約計算の結果をペインバッファ４２に保持させる。そして、ウィンドウレベルサブクエリ処理部４３は、ペインバッファ４２にペイン単位で保持されている中間結果を読み込み、ウィンドウ単位の集約計算を実行する。
【０１１９】
例えば、図１６には、ウィンドウ集約モジュール３４が、ウィンドウ幅（Range）が３分として指定され、スライド幅（Slide）が１分として指定されているウィンドウから属性ｘの最大値を見つけて出力する処理の例が示されている。なお、 図１６に示されている例では、各ウィンドウの終了時刻が、出力タプルのタイムスタンプとして定義されているものとする。例えば、時刻time1から時刻time3までのウィンドウ０では、時刻time1における属性ｘ：７が最大値として出力される。また、時刻time2から時刻time4までのウィンドウ１では、時刻time4における属性ｘ：６が最大値として出力され、時刻time3から時刻time5までのウィンドウ２では、時刻time5における属性ｘ：９が最大値として出力され

る。
【０１２０】
このように、ウィンドウ集約モジュール３４では、入力タプルに対するウィンドウ処理および集約処理を行うことができる。
【０１２１】
ここで、ウィンドウ集約モジュール３４の実装について説明する。図４を参照して上述したように、ウィンドウ集約モジュール３４は、ペインレベルサブクエリ処理部４１、ペインバッファ４２、およびウィンドウレベルサブクエリ処理部４３を備えて構成される。
【０１２２】
初めに、例えば、非特許文献６に開示されているPLQ Control ModuleおよびPLQ Aggregate Moduleを、図３のコンフィギュラブルなハードウェアモジュール２１として実装し、組み合わせることによりペインレベルサブクエリ処理部４１を実現する。次に、非特許文献６に開示されているPane Bufferの実装方法を用いて、ペインバッファ４２を実現する。最後に、非特許文献６に開示されているWLQ Control Module及びWLQ Aggregate Moduleを、図３のコンフィギュラブルなハードウェアモジュール２１として実装し、組み合わせることによりウィンドウレベルサブクエリ処理部４３を実現する。なお、ユーザによって与えられるウィンドウ指定を含む集約クエリからPLQ Control ModuleおよびPLQ Aggregate Module、並びに、WLQ Control ModuleおよびWLQ Aggregate Moduleを実装する方法は、非特許文献６に開示されている。
【０１２３】
そして、クエリ処理部３１において、ペインレベルサブクエリ処理部４１、ペインバッファ４２、およびウィンドウレベルサブクエリ処理部４３は、１つのアグリゲーション・パイプラインとして動作する。図４に示したように、クエリ処理部３１は、複数（Ｎ本）のアグリゲーション・パイプラインを有する。
【０１２４】
また、ウィンドウ集約モジュール３４は、完全に独立して動作することが可能である。これにより、クエリ処理部３１は、複数クエリを完全に並列実行することに対応している。例えば、ユーザは、FPGA２の静的なコンフィギュレーション（図２の第１段階）を行う際に、ウィンドウ集約モジュール３４の数を静的なパラメータとして自由に指定することができる。このように、非特許文献６に開示されている方法を応用することによって、ウィンドウ指定を含む集約クエリにおいて、ウィンドウ幅およびスライド幅の比率に応じてハードウェア・リソース使用量が増加してしまう問題を解決することができる。
【０１２５】
以上のように、CQPH４のアーキテクチャにより実現されるクエリ処理部３１は、クエリ処理部３１を構成する各モジュール（図４に示すブロック）を、それぞれコンフィギュラブルなハードウェアモジュール２１として実装することができる。これにより、クエリ処理部３１は、ランタイム時にユーザによって与えられるクエリに対してインタラクティブに対応することができる。従って、このようなCQPH４を備えるデータストリーム管理システム１は、上述した非特許文献４に開示されている方法に関する２つの重大な欠点を同時に解決することができる。
【０１２６】
なお、本実施の形態では、説明の簡易化のため図４に示したクエリ処理部３１を一例としたが、CQPH４の実現手法を用いて実装可能なアーキテクチャはクエリ処理部３１に限定されるものではなく、本技術は、様々な処理に適用することができる。また、CQPH４のアーキテクチャにおける処理対象として、ストリームデータ以外の様々な形式のデータを用いることができる。さらに、本実施の形態では、説明の簡易化のためにFPGAを用いた実施形態を一例として説明したが、本技術は、FPGAを用いたものに限定されることはない。
【０１２７】
また、本明細書において、「コプロセッサ」という用語は、電子計算機システムにおいてCPU（主処理装置）の補助を行い、計算機の性能向上、計算機の省電力化、計算機のエ

ネルギー消費性能の向上など（ただしこれらに限定されない）に寄与する事を目的とした装置の事である。コプロセッサによって実行されるタスクの範囲は、コプロセッサのアーキテクチャに応じて、固定とすることも、可変とすることもできる。

【０１２８】

また、再構成可能コプロセッサアーキテクチャの例には、幅広い種類の固定型、またはコンフィギュラブルな計算モジュールを実装するように再構成され得るFPGA（Field-Programmable Gate Array）などの再構成可能な論理回路が含まれる。コプロセッサの接続形態としては、SoC（System-on-a-chip）などで見られるような1つの半導体チップ上に直接組み込む方式や、CPU直結型の接続方式、主記憶共有型の接続方式、各種I/Oデバイスと密結合された接続方式、システムバスや入出力バスなどのデータ転送経路上に直接組み込む方式などが含まれる（ただし、これらに限定されるものではない）。

【０１２９】

さらに、DSMSなどのリアルタイムデータ処理基盤におけるコプロセッサとしての利用は、単なる例示的なものであり、本実施の形態は、これに限定されるものではない。また、本技術によって可能とされる高性能のデータストリーム処理には、無数の有用な用途がある。本発明の実施形態は次のものを含む（ただし、これらに限定されるものではない）。

【０１３０】

即ち、各種センサ、組み込み機器、モバイルデバイスなど、データストリームの発生源におけるコプロセッサとしての利用を含む。また、セマンティックルータなど、データストリームの中継装置におけるコプロセッサとしての利用を含む。また、リレーショナルデータベースにおける選択演算や多次元分析に基づくOLAP（Online Analytical Processing）における集約処理など（ただし必ずしもこれらに限定されない）ストリーム的に処理できる計算のハードウェア・アクセラレーションを含む。さらに、その他ストリーム的な処理を行う特定のアプリケーションに特化したアプライアンス製品の一構成要素としての利用を含む。なお、本技術の具体的な実施の形態、CQPHの実現手法、及びCQPHのアーキテクチャについては、上述した説明及び図面を考察すれば当業者には明らかである。

【０１３１】

また、本明細書によって開示された方法では、コンフィギュレーションレジスタの数及びサイズ、並びにプログラマブルなハードウェアブロック内部のロジックに一切制約を設けていない点に留意すべきである。従って、本技術によって開示された方法の幅広い応用が期待される。

【０１３２】

例えば、本実施の形態の一変形例として、プログラマブルなハードウェアブロック内部に、より複雑なロジックを実装し、複数のコンフィギュレーションタプルを用いて複数クロックサイクルでコンフィギュレーションレジスタを書き換えてもよい。また、この場合、コンフィギュレーションレジスタのコピーを複数用意し異なるクエリ構成情報をあらかじめ書き込んでおくことによって、ランタイム時により高速にプログラマブルなハードウェアブロックの動作を切り替えることが可能となる。

【０１３３】

さらに、本明細書において例と説明している技術は、本明細書において具体的には開示しない、任意の１つまたは複数の要素、あるいは１つまたは複数の限定事項がなくても実施可能である。また、本明細書において使用している用語及び表現は、説明のための用語として使用され、限定のための用語として使用されておらず、特許請求の範囲に記載される本技術の範囲において、さまざまな変更が可能であると理解される。従って、当業者であれば、本明細書に開示する方法について変更及び変形を行うことが可能であり、そのような変更及び変形は、特許請求の範囲に包含されると考えられることを理解されたい。

【０１３４】

なお、上述した一連の処理（FPGA２にCQPH４を実装する処理、および、CQPH4にユーザによって与えられるクエリを登録するための処理）は、ハードウェアにより実行することもできるし、ソフトウェアにより実行することもできる。一連の処理をソフトウェアによ

り実行する場合には、そのソフトウェアを構成するプログラムが、専用のハードウェアに組み込まれているコンピュータ、または、各種のプログラムをインストールすることで、各種の機能を実行することが可能な、例えば汎用のパーソナルコンピュータなどに、プログラムが記録されたプログラム記録媒体からインストールされる。

【０１３５】

図１７は、上述した一連の処理をプログラムにより実行するコンピュータのハードウェアの構成例を示すブロック図である。

【０１３６】

コンピュータにおいて、CPU（Central Processing Unit）１０１、ROM（Read Only Memory）１０２、RAM（Random Access Memory）１０３は、バス１０４により相互に接続されている。

【０１３７】

バス１０４には、さらに、入出力インタフェース１０５が接続されている。入出力インタフェース１０５には、キーボード、マウス、マイクロホンなどよりなる入力部１０６、ディスプレイ、スピーカなどよりなる出力部１０７、ハードディスクや不揮発性のメモリなどよりなる記憶部１０８、ネットワークインタフェースなどよりなる通信部１０９、磁気ディスク、光ディスク、光磁気ディスク、或いは半導体メモリなどのリムーバブルメディア１１１を駆動するドライブ１１０が接続されている。

【０１３８】

以上のように構成されるコンピュータでは、CPU１０１が、例えば、記憶部１０８に記憶されているプログラムを、入出力インタフェース１０５及びバス１０４を介して、RAM１０３にロードして実行することにより、上述した一連の処理が行われる。

【０１３９】

コンピュータ（CPU１０１）が実行するプログラムは、例えば、磁気ディスク（フレキシブルディスクを含む）、光ディスク（CD-ROM(Compact Disc-Read Only Memory),DVD(Digital Versatile Disc)等）、光磁気ディスク、もしくは半導体メモリなどよりなるパッケージメディアであるリムーバブルメディア１１１に記録して、あるいは、ローカルエリアネットワーク、インターネット、デジタル衛星放送といった、有線または無線の伝送媒体を介して提供される。

【０１４０】

そして、プログラムは、リムーバブルメディア１１１をドライブ１１０に装着することにより、入出力インタフェース１０５を介して、記憶部１０８にインストールすることができる。また、プログラムは、有線または無線の伝送媒体を介して、通信部１０９で受信し、記憶部１０８にインストールすることができる。その他、プログラムは、ROM１０２や記憶部１０８に、あらかじめインストールしておくことができる。

【０１４１】

なお、本実施の形態は、上述した実施の形態に限定されるものではなく、本開示の要旨を逸脱しない範囲において種々の変更が可能である。

【符号の説明】

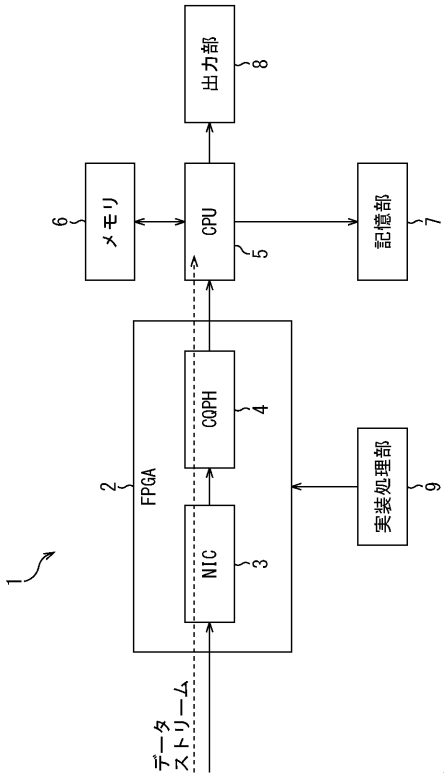
【０１４２】

１　データストリーム管理システム、　２　FPGA、　３　NIC、　４　CQPH、　５　CPU、　６　メモリ、　７　記憶部、　８　出力部、　１１　テンプレート処理部、　１２　論理合成・配置配線処理部、　１３　クエリパーサ、　１４　クエリコンパイラ、　２１　ハードウェアモジュール、　２２　ビットフラグフィールド接続インタフェース、　２３　データフィールド接続インタフェース、　２４　コンフィギュレーションデコーダ、　２５　コンフィギュレーションレジスタ、　２６　プログラマブルハードウェアブロック、　３１　クエリ処理部、　３２　選択モジュール、　３３－１乃至３３－Ｎ　グループ化モジュール、　３４－１乃至３４－Ｎ　ウィンドウ集約モジュール、　３５　統合モジュール、　４１－１乃至４１－Ｎ　ペインレベルサブクエリ処理部、　４２－１乃至４２－Ｎ　ペインバッファ、　４３－１乃至４３－Ｎ　ウィンドウレベルサブクエリ処理部

、　４４－１乃至４４－Ｎ　FIFOバッファ、　６１－１乃至６１－４　選択判断モジュール、　６２－１乃至６２－４　レジスタ、　６３－１乃至６３－９　バイナリレデューサモジュール、　６４－１乃至６４－３　レジスタ、　７１および７２　入力ポート、　７３および７４　第１出力ポート、　７５および７６　第２出力ポート

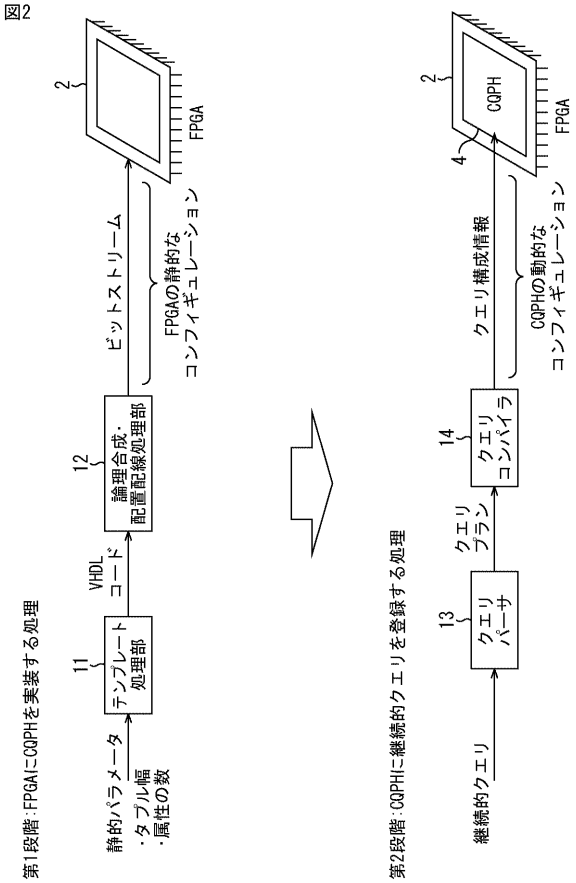
【図１】

図1

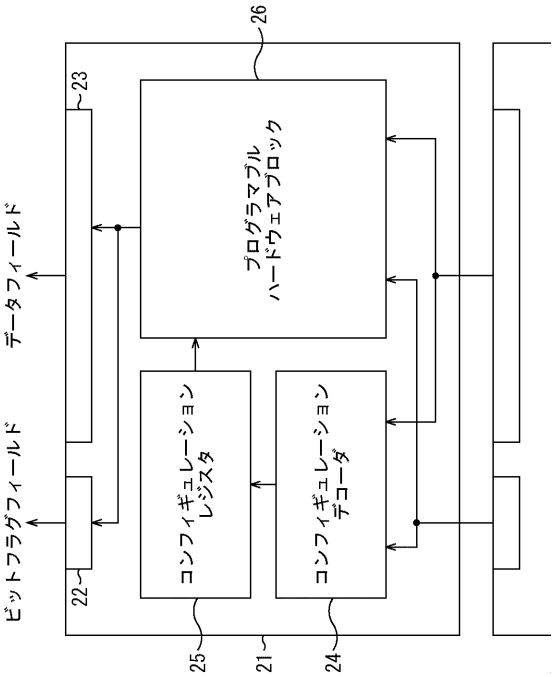


【図２】

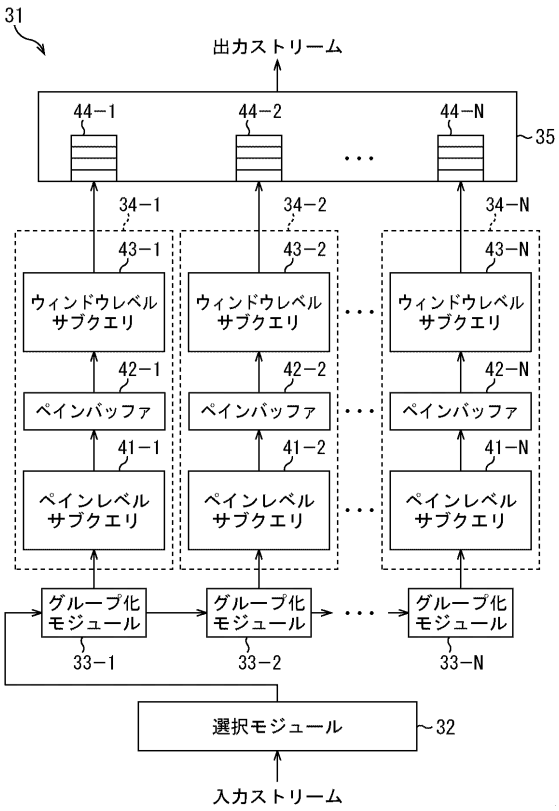
図2



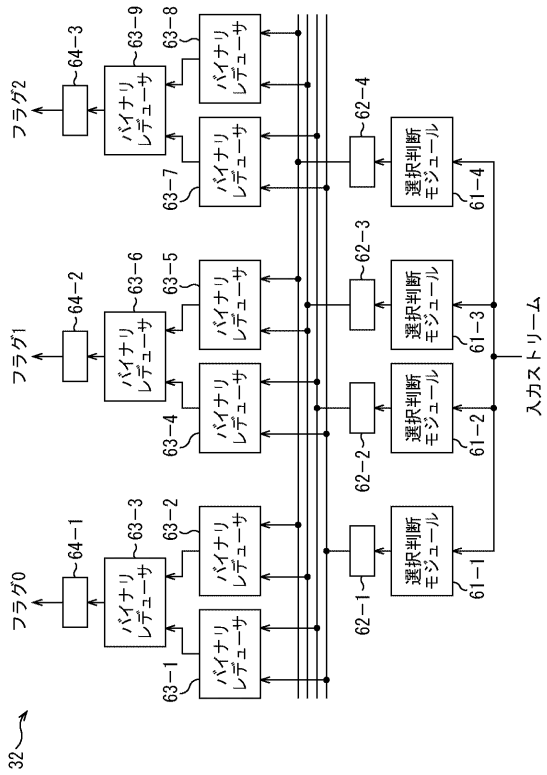
【図３】
図3



【図４】
図4



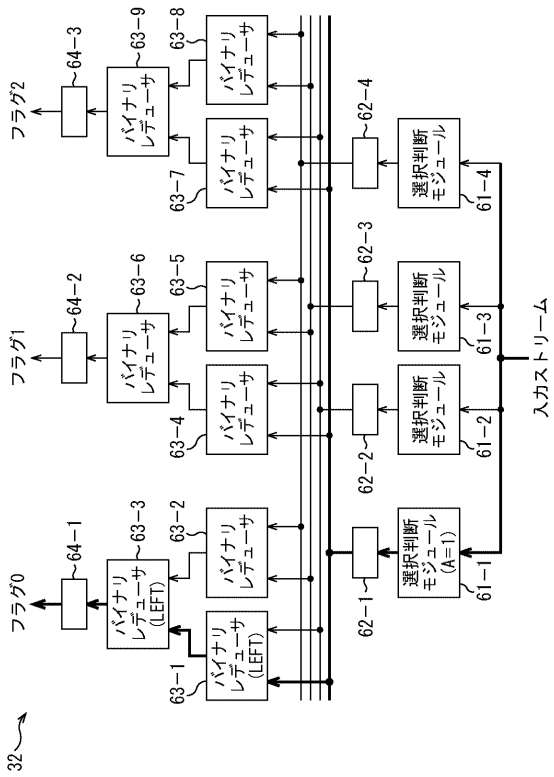
【図５】
図5



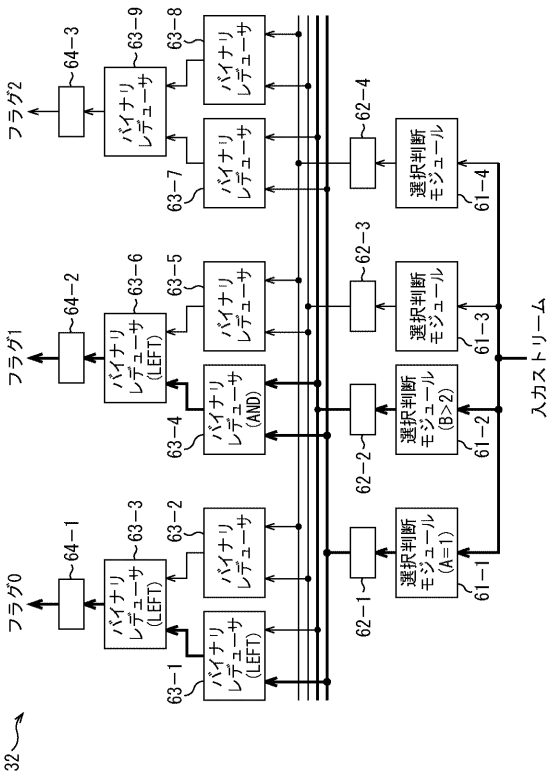
【図６】
図6

クエリQ1	クエリQ2	クエリQ3
SELECT * FROM S WHERE A=1	SELECT * FROM S WHERE A=1 AND B>2	SELECT * FROM S WHERE (A=1 OR B>2) AND C<3

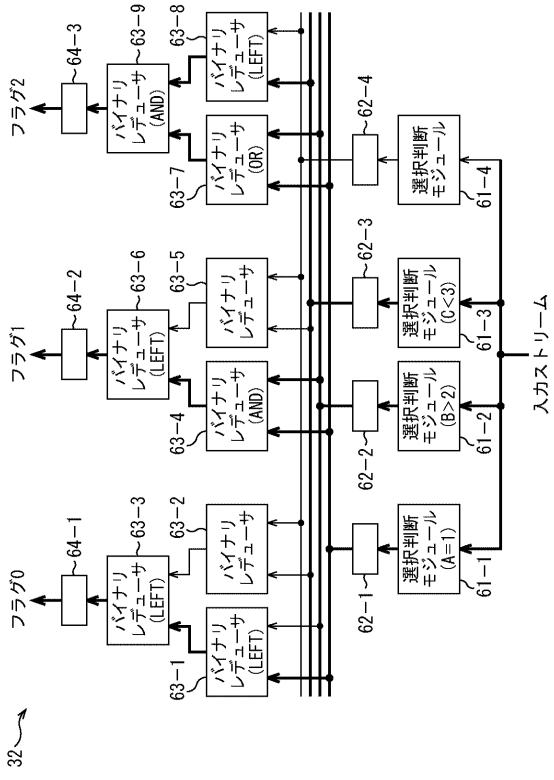
【図７】
図7



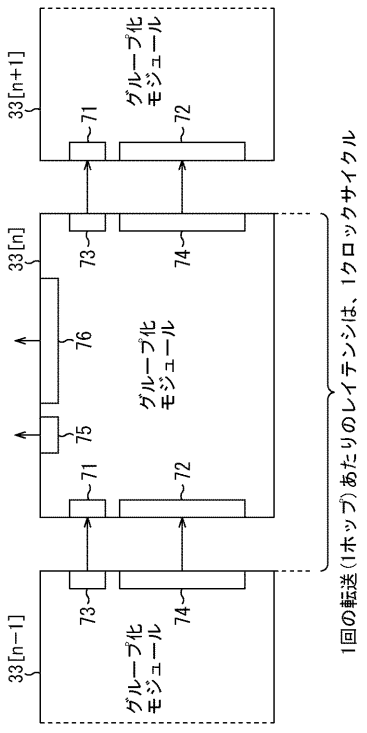
【図８】
図8



【図９】
図9



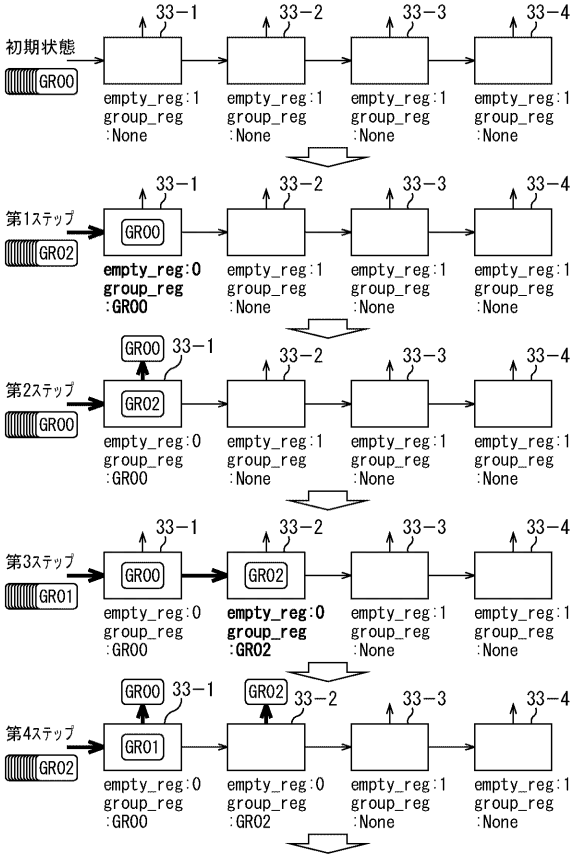
【図１０】
図10



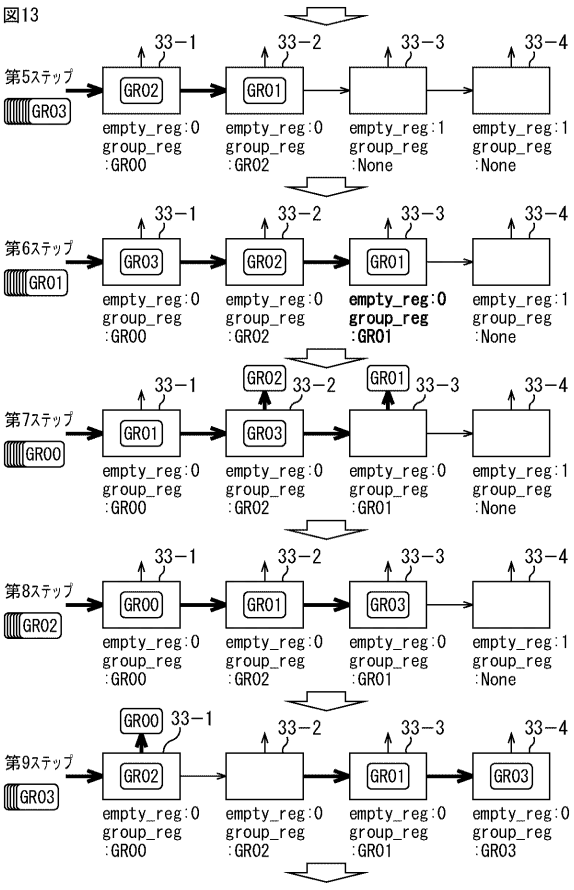
【図１１】
図11

クエリQ4
SELECT item-id, max(bid-price), timestamp FROM bids [RANGE 4 minutes SLIDE 1 minute WATTR timestamp] WHERE item-id >= 1 AND item-id <= 10 GROUP BY item-id

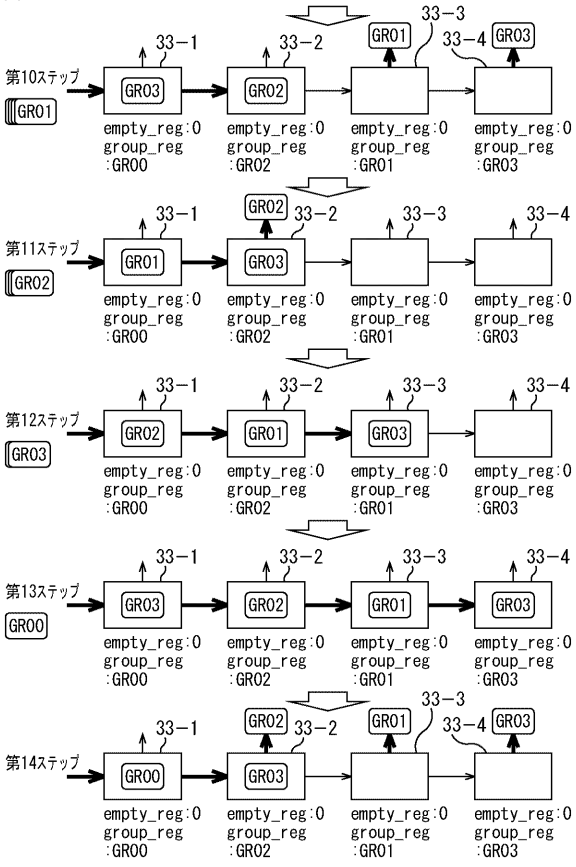
【図１２】
図12



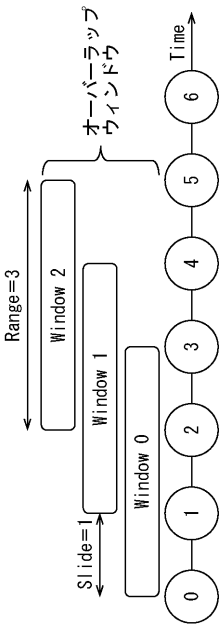
【図１３】
図13



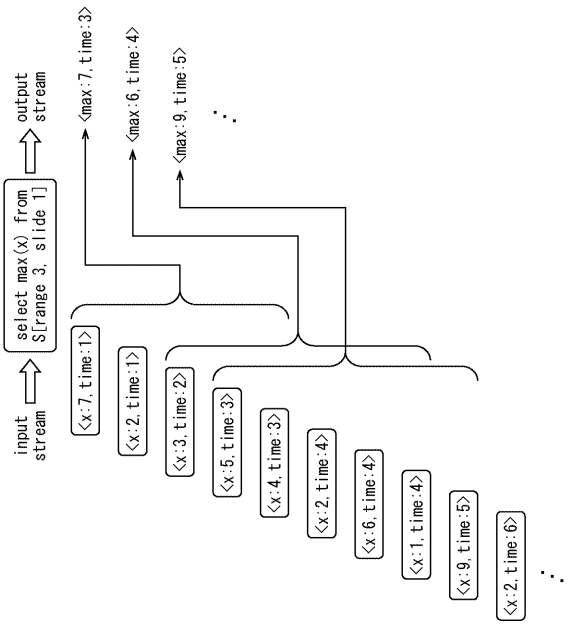
【図１４】
図14



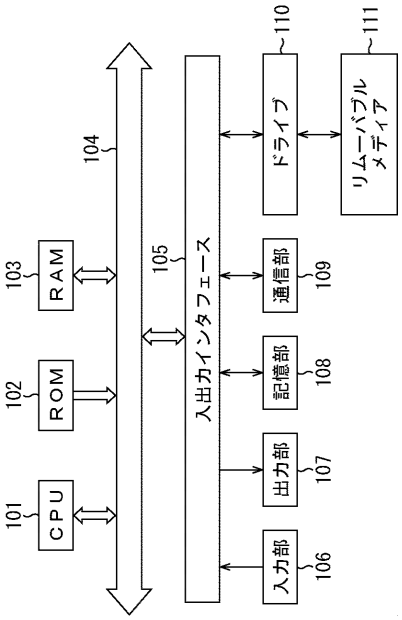
【図１５】
図15



【図１６】
図16



【図１７】
図17



フロントページの続き

(72)発明者　吉見　真聡
東京都調布市調布ケ丘一丁目５番地１　国立大学法人電気通信大学内
(72)発明者　入江　英嗣
東京都調布市調布ケ丘一丁目５番地１　国立大学法人電気通信大学内

Ｆターム(参考) 5B376 AA11　FA21