

(19) 日本国特許庁(JP)

(12) 特 許 公 報(B2)

(11) 特許番号

特許第5110162号

(P5110162)

(45) 発行日 平成24年12月26日(2012.12.26)

(24) 登録日 平成24年10月19日(2012.10.19)

(51) Int.Cl.

F 1

G 0 6 F 17/30 (2006.01)

G 0 6 F 17/30 1 1 O B

G 0 6 F 17/30 3 3 O Z

G 0 6 F 17/30 4 1 5

請求項の数 8 (全 36 頁)

(21) 出願番号	特願2010-518850 (P2010-518850)	(73) 特許権者	000005223
(86) (22) 出願日	平成20年7月1日(2008.7.1)		富士通株式会社
(86) 国際出願番号	PCT/JP2008/061913		神奈川県川崎市中原区上小田中4丁目1番1号
(87) 国際公開番号	W02010/001464	(74) 代理人	100099759
(87) 国際公開日	平成22年1月7日(2010.1.7)		弁理士 青木 篤
審査請求日	平成22年10月25日(2010.10.25)	(74) 代理人	100119987
			弁理士 伊坪 公一
		(74) 代理人	100081330
			弁理士 樋口 外治
		(74) 代理人	100114177
			弁理士 小林 龍
		(72) 発明者	▲葛▼ 毅
			神奈川県川崎市中原区上小田中4丁目1番1号 富士通株式会社内

最終頁に続く

(54) 【発明の名称】 検索装置および検索方法

(57) 【特許請求の範囲】

【請求項 1】

C P Uと、複数の検索コアおよびスケジューラを有するアクセラレータと、を備えた検索装置であって、

前記C P Uは、入力された検索式に従ってオートマトンを作成し、

前記スケジューラは、処理対象テキストを、順番付けしたレコード単位で前記各検索コアに分配し、

前記各検索コアは、前記分配された各レコードをそれぞれ並列的にオートマトン照合し、

前記C P Uは、前記各レコード単位の照合結果を該各レコードの前記順番付けの順で論理式評価することを特徴とする検索装置。

10

【請求項 2】

請求項 1 に記載の検索装置において、さらに、

前記C P Uおよび前記検索コアの共有データを格納するローカルメモリを備え、

該共有データは、前記複数のレコードの照合結果を格納する照合結果情報と、複数のレコードを管理する管理情報とを有し、

前記各レコードの前記順番付けは、前記管理情報として前記ローカルメモリに保持されることを特徴とする検索装置。

【請求項 3】

請求項 2 に記載の検索装置において、

20

前記照合結果情報は、前記各検索コアに対して前記ローカルメモリの2枚以上の記憶領域に分割して割り当てて管理されることを特徴とする検索装置。

【請求項4】

請求項2または3に記載の検索装置において、

前記スケジューラは、前記管理情報を読んで空いている前記検索コアに前記順番付けの順で前記管理データを分配して照合を指示し、検索コアはランダムに照合完了することを特徴とする検索装置。

【請求項5】

請求項1～4のいずれか1項に記載の検索装置において、

前記処理対象テキストは、構造化文書であり、

前記各検索コアは、前記構造化文書のデータから所定の文字列を検索することを特徴とする検索装置。

【請求項6】

CPUと、複数の検索コアおよびスケジューラを有するアクセラレータと、を備えた検索装置を使用した検索方法であって、

前記CPUが、入力された検索式に従ってオートマトンを作成し、

前記スケジューラが、処理対象テキストを順番付けされたレコード単位で前記各検索コアに分配し、

前記各検索コアが、前記分配された各レコードをそれぞれ並列的に照合処理し、

前記CPUが、前記各レコード単位の照合結果を該各レコードの前記順番付けの順で論理式評価することを特徴とする検索方法。

【請求項7】

請求項6に記載の検索方法において、

前記スケジューラは、前記CPUおよび前記検索コアの共有データにおける複数のレコードを管理する管理情報を読んで空いている前記検索コアに前記順番付けの順で前記管理データを分配して照合を指示し、検索コアはランダムに照合完了することを特徴とする検索方法。

【請求項8】

CPUと、複数の検索コアおよびスケジューラを有するアクセラレータと、を備えたコンピュータに、

前記CPUに、入力された検索式に従ってオートマトンを作成させる手順と、

前記スケジューラに、処理対象テキストを順番付けされたレコード単位で前記各検索コアに分配させる手順と、

前記各検索コアに、前記分配された各レコードをそれぞれ並列的に照合処理させる手順と、

前記CPUに、前記各レコード単位の照合結果を該各レコードの前記順番付けの順で論理式評価させる手順と、を実行させる検索プログラム。

【発明の詳細な説明】

【技術分野】

【0001】

この出願は、検索装置および検索方法に関し、特に、オートマトン処理技術を適用して与えられた入力文字列であるテキストから任意の文字列であるパターンを検索する検索装置および検索方法に関する。

【背景技術】

【0002】

近年、さまざまな分野で情報量が爆発的に増加している。そして、分野によっては、情報量がギガバイトオーダーからテラバイトオーダーになって、その莫大なデータから所望のデータを短時間で取り出すことが困難となって来ている。

【0003】

与えられたテキストから任意の種類のパターンを検索するパターンマッチングは、ワー

10

20

30

40

50

プロソフトやデータベースの検索など様々な分野で応用されている。

【0004】

すなわち、近年、XMLデータベースをはじめとするパターン照合（検索）処理が必要なアプリケーションの需要が増加傾向にあり、その高速化のためにパターン照合を行うオートマトン照合装置（検索コア）をプロセッサに搭載する手法が提案されている。

【0005】

そして、従来、このような照合装置を複数有するプロセッサシステムにおいては、1枚の照合すべきテキストを如何にして高速に照合するかが検討されている。

【0006】

具体的に、従来、照合装置を複数有するプロセッサシステムを使用して、複数文字を一度に照合する方法が提案されている。

10

【0007】

さらに、従来、照合装置を複数有するプロセッサシステムを使用して並列処理を実行するために、照合テキストを分割する方法も提案されている。

【0008】

【特許文献1】特開2005-242672号公報

【特許文献2】特許第2865831号公報

【発明の開示】

【発明が解決しようとする課題】

【0009】

20

従来、照合装置を複数有するプロセッサシステムを使用してテキストの照合を行うものが提案されている。

【0010】

しかしながら、例えば、データベースがはじめから複数のXMLテキスト群により構成されたXMLデータベースシステムの照合（検索）処理において、はじめからパターン照合を行うと各XMLテキスト間で並列性が高いため、照合装置の搭載数を増やすなどして照合処理を容易に並列処理して高速化することができる。しかしながら、その照合結果を用いたCPUによる後処理の比重は大きくなる。

【0011】

また、照合処理とCPUの後処理とを並列動作させる場合、そのオーバーヘッドを減らすためのデータの受け渡し方法および制御方式が、検索（照合）システム全体の高速化を実現するのに重要となる。

30

【0012】

さらに、各照合結果のテキスト単位の順序関係を保持してCPUで後処理を行う仕組みも必要となる。例えば、XMLデータベースシステムにおける検索において、もし、XMLテキストのCPU処理順序がXMLデータベースで管理されている順序と毎回異なると、検索結果が毎回違った順番に表示され、ユーザにとって使いにくいシステムとなってしまう。

【0013】

そのため、並列動作により順不同に照合完了した照合結果から、レコード順に後処理可能となった段階で並列に処理するための効率的な仕組みも必要となる。

40

【0014】

本出願は、上述した従来および関連技術における課題に鑑み、各照合装置（検索コア）とCPU処理の並列動作のオーバーヘッドを減らしてデータの受け渡しを行いシステムの高速化を行うことが可能な検索装置および検索方法の提供を目的とする。

【課題を解決するための手段】

【0015】

第1の実施形態によれば、CPUと、複数の検索コアおよびスケジューラを有するアクセラレータと、を備えた検索装置が提供される。前記CPUは、入力された検索式に従ってオートマトンを作成し、前記スケジューラは、処理対象テキストを、順番付けしたレコ

50

ード単位で前記各検索コアに分配する。さらに、前記各検索コアは、前記分配された各レコードをそれぞれ並列的にオートマトン照合し、前記CPUは、前記各レコード単位の照合結果を該各レコードの前記順番付けの順で論理式評価する。

【0016】

第2の実施形態によれば、CPUと、複数の検索コアおよびスケジューラを有するアクセラレータと、を備えた検索装置を使用した検索方法が提供される。前記CPUが、入力された検索式に従ってオートマトンを作成し、前記スケジューラが、処理対象テキストを順番付けされたレコード単位で前記各検索コアに分配する。さらに、前記各検索コアが、前記分配された各レコードをそれぞれ並列的に照合処理し、前記CPUが、前記各レコード単位の照合結果を該各レコードの前記順番付けの順で論理式評価する。

10

【0017】

第3の実施形態によれば、CPUと、複数の検索コアおよびスケジューラを有するアクセラレータと、を備えたコンピュータに、前記CPUに、入力された検索式に従ってオートマトンを作成させる手順と、前記スケジューラに、処理対象テキストを順番付けされたレコード単位で前記各検索コアに分配させる手順と、前記各検索コアに、前記分配された各レコードをそれぞれ並列的に照合処理させる手順と、前記CPUに、前記各レコード単位の照合結果を該各レコードの前記順番付けの順で論理式評価させる手順と、を実行させる検索プログラムが提供される。

【発明の効果】

【0018】

20

各実施形態によれば、各照合装置とCPU処理の並列動作のオーバーヘッドを減らしてデータの受け渡しを行いシステムの高速化を行うことが可能な検索装置および検索方法を提供することができる。

【発明を実施するための最良の形態】

【0019】

まず、検索装置および検索方法の実施例を詳述する前に、従来技術およびその問題点を、図面を参照して説明する。

【0020】

図1Aおよび図1Bはオートマトンによるキーワード検索アルゴリズムの一例であるシグマ(Sigma)アルゴリズムを説明するための図である。なお、図1Aおよび図1Bは、検索対象の文書から"blue", "green", "red"および"yellow"のキーワードが存在するか否かを検索する例を示している。

30

【0021】

まず、図1Aに示されるように、各キーワード条件に対応したオートマトンを作成する。具体的に、"0"で示す根から、各キーワードの先頭文字"b", "g", "r", "y"に遷移し、さらに、各キーワード列の文字に遷移するオートマトンを作成する。

【0022】

各キーワード列の最後の文字まで一致すれば、そのキーワード列が検索されたことになる。ここで、例えば、"green"のキーワード列には、"re"が含まれており、その後に"d"があれば"red"と一致するので、図1Aに示されるように、"green"の途中から"red"の"d"に遷移する経路が存在する。

40

【0023】

検索は、根から始まり、オートマトン内のキーワードがあれば順次遷移し、キーワードと関係のない文字が読み込まれたら根に戻る。

【0024】

例えば、図1Bに示されるように、"black"を含む文書が入力されると、"blue"の先頭の"b"に遷移し、さらに次の"l"に遷移するが、次が"a"であるため根に戻る。

【0025】

このように、文書内にオートマトンのキーワード列があれば、各キーワード列の最後ま

50

で到達してヒット (Hit) 情報が出力され、キーワード列が存在することが判明する。ここで、オートマトン検索において、各キーワード列の何番目の文字にあるかをノードまたは「状態」で表す。例えば、根が状態 1 で、"blue" の "b" にいれば状態 2 で、"u" にいれば状態 4 と称する。

【0026】

図 2 は検索装置の一例を概略的に説明するための図であり、決定性有限オートマトンを用いた検索装置を説明するためのものである。

【0027】

図 2 に示されるように、ディスク装置などに格納された検索対象文書 11 が流し込まれると、そこから順に文字バイトコード 12 が取り出される。以下の説明では、文字バイトコード 12 が 8 ビットで、256 エントリを形成するものとする。

10

【0028】

レジスタに保持された現在の状態 (現在の状態を示すデータ) 13 と、文字バイトコード 12 とを合わせたインデックスを、検索キーワードオートマトンを形成するメモリ 15 の入力アドレス 14 とする。

【0029】

入力アドレス 14 が入力されたメモリ 15 は出力データ 16 を出力するが、この出力データには、次の状態 17 およびヒット情報 (ヒット値) が含まれる。そして、次の状態 17 は、現在の状態 13 に置き換わることになる。

【0030】

20

図 3 は従来の検索装置における検索シーケンスの一例を概略的に示す図であり、図 4 は従来の検索方法による処理順を概略的に示す図である。

【0031】

図 3 に示されるように、従来の検索装置における検索シーケンスは、例えば、検索式が入力される (30) と、オートマトンが作成され (31)、そのオートマトンを使用してレコードの照合 (32a, 32b, ...) および論理式評価 (33a, 33b, ...) が繰り返される。

【0032】

ここで、オートマトンの作成 31, レコードの照合 32a, 32b, ... および論理式評価 33a, 33b, ... は、全て CPU (処理装置) で処理される。

30

【0033】

さらに、図 4 に示されるように、従来の検索方法による処理順は、オートマトンを使用したレコードの照合→照合結果→論理式評価→検索結果をレコード順に行わなければならない。

【0034】

すなわち、従来の検索方法による処理順は、レコード a の照合 (1) を行った後、レコード a の評価 (2) を行い、さらに、レコード b の照合 (3) を行った後、レコード b の評価 (4) を行い、そして、レコード c の照合 (5) を行った後、レコード c の評価 (6) を行っていた。そして、検索結果は、レコード順に維持する必要があった。

【0035】

40

従来の検索装置および検索方法では、各照合装置と CPU 処理の並列動作のオーバーヘッドが多くなってシステム動作が遅くなり、或いは、検索結果をレコード順に維持しなければならなかった。

【0036】

以下、検索装置および検索方法の実施例を、添付図面を参照して詳述するが、その前に、一例として、XML データベース検索用アクセラレータ (AC) について説明する。

【0037】

検索用データベースソフトウェアは、例えば、IA サーバでは対応困難な大規模 XML データベースを必要とするユーザ向けに、高性能プロセッサを搭載した専用サーバマシンに移植され、この専用サーバ向けプロセッサとしては、XML レコード照合処理を行うア

50

クセラレータ（ＡＣ）を搭載して処理の高速化の図るようになっている。

【００３８】

まず、本実施形態の検索装置（プロセッサシステム）に適用されるアクセラレータ（スケジューラＳＤおよび検索コア（照合装置）ＳＤ）の概要を説明する。すなわち、アクセラレータの概要、ブロック構成および動作を述べ、さらに、各モジュールの役割を説明する。

【００３９】

本実施形態の検索装置は、例えば、ＸＭＬデータベースにおける文字列の検索を行うものであり、そのデータベースをユーザから与えられた検索式にマッチするデータを検索する処理である。

10

【００４０】

使用されている検索アルゴリズムでは、例えば、与えられた検索式に対応するオートマトンを作成し、ＸＭＬデータの管理単位であるレコードをそのオートマトンの入力にして流し込むことで照合を行うようになっている。

【００４１】

本実施形態の検索装置における検索処理は大きくレコード照合と論理式評価の２つに分けられる。アクセラレータは、このうちのレコード照合処理を担当する。

【００４２】

図５は本実施形態の検索装置における検索シーケンスの一例を概略的に示す図である。

図５と前述した図３との比較から明らかなように、本実施形態の検索装置における検索シーケンスでは、レコード照合３２ａ，３２ｂ，…と論理式評価３３ａ，３３ｂ，…を全てＣＰＵで処理するのではなく、スケジューラＳＤおよび複数のオートマトン処理装置（例えば、４個の検索コア）ＳＣを有するアクセラレータＡＣにより処理する。

20

【００４３】

すなわち、スケジューラＳＤは、レコード分配３４ａ，３４ｂ，…を行い、また、検索コアＳＤは、オートマトンの登録３５およびレコード照合３２を行う。このように、アクセラレータＡＣとしてスケジューラＳＤおよび複数の検索コアＳＣを搭載することにより、レコード照合自体を並列処理して高速化し、さらに、論理式評価と並列に動作させるようになっている。

【００４４】

30

図６は本実施形態の検索方法による処理順を概略的に示す図である。

図６と前述した図４との比較から明らかなように、まず、参照符号６１において、アクセラレータＡＣの４個の検索コアＳＣによりレコードの照合を４つ並列的に処理（１）する。また、参照符号６２において、レコード順を保持する配列と照合結果を保持するバッファという構成にすることで、照合順のレコード維持から開放し、並列処理を実現する。なお、参照符号６３のように、論理式評価は、ＣＰＵによりＸＭＬレコードの順に逐次処理する必要がある。

【００４５】

図７は検索装置の一実施例の要部を示すブロック図である。

図７に示されるように、メインメモリに格納されているＸＭＬデータの各レコード（ＸＭＬデータベースはレコードと呼ばれる単位で保持されている）を各検索コアＳＣが自立的にメインメモリにアクセスして読み出して照合を行い、照合結果であるヒット情報をＣＰＵから高速にアクセスできるＬ２キャッシュまたはワークメモリ（WorkRAM）に置かれる共有データに書き込む。

40

【００４６】

ＣＰＵとアクセラレータＡＣで共有される制御データは、大きく２つの配列に分かれる。その１つは、レコード情報（レコードの先頭ポインタなど）を保持するレコードソート配列で、他の１つは、照合結果を保持するイベントバッファ配列である。

【００４７】

スケジューラ（スケジューラ回路）ＳＤは、各検索コアＳＣから通知される空状況をモ

50

ニタして、空いた検索コアにレコード情報を渡し、照合を指示する。CPUはヒット情報を用いてアクセラレータACと並列に論理式評価を行う。

【0048】

図8は検索装置の他の実施例の要部を示すブロック図である。

図8に示す実施例は、上述した図7の構成を2つ設けたもので、これを1つのチップとして構成するようになっている。

【0049】

図8に示されるように、本実施例では、1つの半導体チップ(LSI)CHIPが、図7に示すCPU、アクセラレータACおよびL2キャッシュを2組有している。すなわち、半導体チップCHIPは、2つのCPUシステムPSを備え、各CPUシステムPSは、CPUコア、スケジューラSDおよび複数の(図では4個の)検索コアSCを有するアクセラレータACおよびL2キャッシュ(または、Work RAM)を備えている。

10

【0050】

なお、図7および図8の構成は、単なる例であり、例えば、1つの半導体チップCHIPが、CPUと8個の検索コアSCおよびスケジューラSDを有するアクセラレータACとを2組、或いは、4組等有するよう構成してもよい。

【0051】

図9は各実施例が適用される検索サーバの一例を概略的に示す図であり、また、図10は図9の検索サーバを含む検索システムの全体構成を概略的に示す図である。図9および図10において、参照符号SSは検索サーバ、RSはラックシステム、PBはプロセッサボックス、PMはプロセッサモジュール、PSはプロセッサシステム(検索装置)を示している。

20

【0052】

図9では、検索サーバSSは、1つのラックシステムRSで構成されており、そのラックシステムRSは、複数個(例えば、15個)のプロセッサボックスPBを備え、さらに、各プロセッサボックスPBは、それぞれ複数枚(例えば、28枚)のプロセッサモジュールPMを備えて構成されている。

【0053】

また、各プロセッサモジュールPMは、それぞれ3つの半導体チップCHIPと3つのメインメモリを備え、さらに、各半導体チップCHIPは、各々が1つのCPUと、4つの検索コアSCおよびスケジューラSDを有するアクセラレータACで構成される2つのプロセッサシステムPSを備えて構成されている。

30

【0054】

ここで、照合処理は、例えば、各プロセッサシステムPSに設けられた各検索コアSCにより各レコード単位でランダム(アウトオブオーダー)に実行することができるが、論理式評価は、例えば、CPUによりXMLデータのレコード順に行うことになる。

【0055】

なお、各検索コアSCには、例えば、それぞれ高速アクセスが可能なキャッシュメモリ(一時キャッシュ)が内蔵されており、このキャッシュメモリに対してオートマトンを作成して格納するようになっている。

40

【0056】

図10に示されるように、データ管理部は、ディレクタサーバDSおよびストレージサーバSTRで構成され、また、検索処理部は、検索サーバSSで構成される。検索サーバSSは、例えば、LAN等のネットワークNETを介してディレクタサーバDSに繋がれ、例えば、そのディレクタサーバDSに繋がれたXMLのような構造化文書のデータが格納されたストレージサーバSTRにおける文書データを処理する。

【0057】

このように、例えば、図9に示す検索サーバでは、2つのCPUコア×(システムのチップ数)個のプロセッサがXMLデータに含まれる全てのレコードをプロセッサ個数に分割してそれぞれ担当して別々に検索処理を行う。

50

【0058】

具体的に、例えば、CPUが合計256個、レコードがレコードIDの0000からFFFFまでである場合、レコードIDの0000～00FF, 0100～01FF, …というように256個に分割し、各プロセッサがそれぞれ担当することになる。

【0059】

また、図10に示す検索システムにおいて、システムが起動されると、ストレージSTRに格納されているXMLデータベースは、ディレクタサーバDSにより分割され、検索サーバSSの各プロセッサ(CPU)に分配されて対応するメモリ上に格納される。

【0060】

検索が始まると、まず、検索式が検索サーバSSに送られて検索サーバSSがオートマトンを作成し、各アクセラレータACに格納される。そして、各CPUは、自分の担当するXMLデータベースの断片をアクセラレータACに流して照合する。

10

【0061】

なお、図9および図10に示す検索装置および検索システムは単なる例であり、適用されるシステムの規模や要求される検索性能等に応じて様々に変化され得るのはいうまでもない。例えば、より大規模なシステムでは、検索サーバSSを複数台のラックシステムRSで構成し、また、複数台のディレクタサーバDSにより管理を行うといった構成にすることも可能である。

【0062】

図11はレコードIDからレコードを特定する処理を説明するための図であり、XMLデータのデータ構造を示すものである。

20

【0063】

図11に示されるように、各レコードは、XMLデータテーブルXDT、レコードインデックステーブルRITおよびメモリデータ情報MDIと呼ばれる構造体により管理される。

【0064】

まず、XMLデータテーブルXDTには、XMLデータ全体の情報が収められており、レコードインデックステーブルRITへのポインタが含まれている。

【0065】

レコードインデックステーブルRITは、メモリデータ情報MDIへのポインタ配列となっており、レコードIDの上位2バイトにより要素が識別される。

30

【0066】

メモリデータ情報MDIは、各レコードのレコード情報をメンバに持つ構造体配列であり、レコードIDの下位2バイトにより要素が識別される。なお、レコード情報には、アクセラレータACで照合を行う単位となるレコードへのポインタが含まれている。

【0067】

なお、図6を参照して述べたように、照合処理は各レコード単位でランダム(アウトオブオーダー)に実行可能であるが、論理式評価はこのXMLデータのレコード順に行う必要がある。

【0068】

図12は各モジュールの役割を説明するための図である。

40

図12に示されるように、モジュールは、大きくソフトとハードに分類される。さらに、ソフトは、アプリ(アプリケーションプログラム)およびドライバに分類され、また、ハードは、スケジューラSDおよび検索コアSCに分類される。

【0069】

アプリ(Sigma Evaluation: シグマ評価)は、論理式評価を行い、バッファフルを検出したら、シグマバッファフルを呼ぶ。

【0070】

ドライバは、シグマ検索(Sigma Search)およびシグマバッファフル(Sigma Buffer Full)に分類される。シグマ検索は、XMLのインデックスを参照して、レコードへのポ

50

インタをインオーダで取り出し、ソート配列に登録する。適当なタイミング（ポーリング）でバッファ状態を見て、論理式評価関数を起動する。また、論理式評価関数が終了したら、そこまでのソート配列エントリをフラッシュする。そして、適当なタイミング（論理式評価関数が終了したら評価終了レコード個数をカウントし、閾値を設けて起動など）でレコード情報をソート配列に補充する。

【0071】

シグマバッファフルは、論理式評価がイベントバッファ終端に到達したら、呼ばれる。そして、イベントバッファフラッシュを行って、現在論理式評価中のレコードの照合結果が終了または次のイベントバッファがフルになるまで待って、終了する。

【0072】

スケジューラSDは、アイドル状態の検索コアSCがあれば、ソート配列に登録された照合未開始の先頭のレコード情報（ポインタなど）を検索コアSCに分配して、検索開始を指示する。

【0073】

検索コアSCは、レコード情報が与えられたら、メモリからレコードの文字列を読み込んで照合を行う。ヒットしたらイベントバッファに書き込む。イベントバッファを跨ぐなどがあった場合は、バッファ状態フラグを適切に変更する。そして、照合が終了したら、スケジューラSDにアイドル状態を通知して停止する。イベントバッファフルなどがあったらストールして、イベントバッファがフラッシュされるのを待つ。

【0074】

次に、アクセラレータACのドライバソフトウェアとアプリケーションの関係について説明する。

【0075】

図13は各API関数と呼び出し手続きを説明するための図であり、アプリケーション、API関数、ドライバ（アクセラレータライブラリ138）およびアクセラレータACの関係を示している。

【0076】

図13に示されるように、まず、検索式が与えられると、アプリケーション側で対応するオートマトンを作成（131）する。さらに、シグマロックハンドル（Sigma Alloc Handle）関数およびシグマコンフィグ（Sigma Config）関数を呼んで、アクセラレータACの初期設定（作業域取得132および環境設定133）を行う。なお、シグマコンフィグ関数は、システム起動時のみとしてもよい。

【0077】

そして、シグマ登録（Sigma Registration）関数を呼んで作成したオートマトンをアクセラレータACに登録（134）する。さらに、シグマ検索（Sigma Search）関数を呼んで検索処理の本体を開始（照合実行を起動：135）する。このシグマ検索関数で検索の全体が実行され、アプリケーションは終了を待つ。

【0078】

最後に、シグマフリーハンドル（Sigma Free Handle）関数を呼んで一連の検索処理が終了（作業域を解放136）する。なお、シグマ検索関数は、レコードの照合をアクセラレータACに指示し、論理式評価の準備ができたレコードを、シグマ評価（Sigma Evaluation）関数を呼んで論理式評価（137）を行う。また、シグマバッファフル（Sigma Buffer Full）関数は、照合結果を保持するイベントバッファが切り替わる際に呼ばれる。

【0079】

次に、本アクセラレータACの制御方式について説明する。

はじめに、制御データであるレコードソート配列とイベントバッファ配列について、詳細を述べる。さらに、アクセラレータの各動作について、主な動作である検索動作（Sigma Search関数に対応）について述べ、全体制御の遷移概要について述べる。そして、残りのオートマトン登録動作（Sigma Registration関数に対応）および環境設定動作（Sigma Config関数に対応）について述べる。

【0080】

図14は制御データの詳細なデータ構造を示す図である。

まず、図14に示されるように、制御データは、各レコードの情報を管理するレコードソート配列と、検索コアによる照合結果を書き込むイベントバッファ配列との2つに大きく分かれている。

【0081】

レコードソート配列の各行には、XMLデータの各レコードがXMLデータのインデックス順に格納され、照合と論理式評価が終了したレコードの行から解放され、そして、行が終端まで使用されたら循環して先頭に戻って使用される。

【0082】

ここで、各項目は、次の意味を有する。

・照合状態フラグ：そのレコードの検索コアによる照合状態が記録され、論理式評価可能の判定、および、バッファフラッシュ時の論理式評価再開可能の判定に使用される。なお、フラグの意味は、次の通りである。

【0083】

0：照合未終了かつ照合がイベントバッファを跨いでいない。

1：照合が1つでもイベントバッファを跨いだ。

2：照合が終了した。

・検索コア番号：そのレコードを照合し、または、照合した検索コアの番号で、バッファフラッシュ時に使用される。

【0084】

・*レコード：XMLデータのレコードの先頭アドレスへのポインタで、照合開始時に検索コアに渡されて使用される。なお、以下において、「*」はアドレスを示している。

【0085】

・有効フラグ：その行が使用されているかどうかのフラグで、スケジューラがその行を自分のバッファ内に持ってきたときに有効かどうかのチェックに使用される。

【0086】

0：未使用。

1：使用。

・レコードID (Record ID)：論理式評価で使用され、本発明の制御には直接関与しない。

【0087】

・*イベントバッファ：照合結果が格納されているイベントバッファ領域へのポインタで、論理式評価で使用される。

【0088】

なお、「*登録top」および「*登録bottom」の2つのポインタ変数は、現在使用されている行の先頭アドレスおよび終端アドレスを指し示し、ドライバ関数シグマ検索 (Sigma Search) により保持される。次検索topポインタ変数は、現在登録されているレコードの中でまだ照合開始していない行の先頭アドレスを示し、スケジューラ内レジスタに保持されている。

【0089】

イベントバッファ配列には、検索コアによる照合結果が書き込まれる。なお、図14は、検索コアが4個 (検索コア0, 1, 2, 3: SC) で、各検索コアに割り当てられた領域がイベントバッファ4枚 (A, B, C, D) の場合を示している。

【0090】

各イベントバッファの終端エントリには、次イベントバッファ (例えば、AならばB、BならばC、…) へのポインタが予め書き込まれている。そして、例えば、図14における検索コア0のイベントバッファに示されるように、検索コアおよび論理式評価関数 (Sigma Evaluation: シグマ評価) が、それぞれ自分で次のバッファに移行できるようになっている。

【0091】

図14において、荒い点線RETは、論理式評価で現在参照する最後のエントリの位置を示し、また、細かい点線WETは、照合コアで現在照合結果を書き込んでいる最後のエントリWETを示す。さらに、太線NRは、そのレコードがまだ照合終了していないことを示す。なお、バッファ枚数は、実際には2枚を想定している。

【0092】

この他に制御に必要なデータとしてバッファ距離がある。これは、検索コアと論理式評価で現在使用されているイベントバッファ間の距離を示し、例えば、図14の検索コア0では『4』が書き込まれている。このデータは、検索コア内部のレジスタに格納されており、検索コアがイベントバッファを跨ぐごとに『1』を加え(+1) また、イベントバッファがフラッシュされるごとに『1』を引く(-1)。

10

【0093】

上記のデータは、検索コアの動作のみでなくバッファフラッシュ時の動作判定にドライバ(CPU側)からも参照されるため、そのコピーがスケジューラ(SD)に置かれたマップIOレジスタにもコピーされる(或いは、検索コアのバッファ距離レジスタがメモリマップ上に置かれる)。

【0094】

図15は共有データの構造(制御データの配置)を説明するための図である。

図15に示されるように、レコードソート配列およびイベントバッファ配列は、L2キャッシュまたはワークRAM(WorkRAM)領域に置かれる。

20

【0095】

バッファ距離のオリジナルは、検索コア(SC)内部に置かれ、バッファ距離のコピーは、スケジューラ(SD)に置かれる。次照合topは、スケジューラ内部におかれる。また、図15では、後で検索動作を説明するためのスケジューラと検索コアSC内にあるレジスタも描かれている。

【0096】

ここで、スケジューラは、次の意味を持っている。

- ・アイドル(idle)フラグ：各検索コアの状態を示す。

0：アイドル状態。

1：ビジー状態。

30

- ・*次検索top：次に照合するレコード情報が格納されているソート配列のエントリアドレスを指すレジスタ。

【0097】

- ・レコード情報バッファ：レコードソート配列のエントリを一時的に格納するバッファであり、検索コアの個数分だけある。*次照合topからバッファ数分だけ持ってくるが、もし、ソート配列の有効フラグが『0』であれば破棄して、*次照合topは適当なサイクル後に再度読み込む。

【0098】

- ・バッファ距離コピー：検索コアのバッファ距離のコピーを置く。
- ・フラッシュ完了フラグ：イベントバッファフラッシュの待ち合わせに使う。
- ・*登録bottomコピー：*登録bottom変数のコピーを保持する。スケジューラがレコード情報をバッファリングする際に、次照合topが追い越さないために使用する。

40

【0099】

また、検索コアは、次の意味を持っている。

- ・文字バッファ：照合するレコードを一時的に格納する数十バイト程度のバッファで、検索コアは、ここから1文字ずつ使用し、オートマトンを遷移させて照合する。

【0100】

- ・*イベントバッファ：各照合コアがレコード照合のヒット情報を書き込む先のイベントバッファへのポインタである。

50

【0101】

・バッファ距離：論理式評価と照合の同期に使うもので、その検索コアのイベントバッファにおける現在の論理式評価読み出し位置と照合結果書き込み位置のバッファ枚数単位の距離である。なお、初期値は『0』で、書き込みは検索コアのみが行う。照合結果書き込み時に現在のバッファ終端に到達したタイミングで『1』を加え（+1）また、イベントバッファフラッシュを指示されると『1』を引く（-1）。

【0102】

図16および図17は、各データの書き込みおよび読み出しの関係を説明するための図である。

【0103】

図16に示されるように、まず、各データの書き込みに関して、シグマ検索（Sigma Search）からレコードソート配列への書き込みが多いが、これは、シグマ検索でXMLデータインデックスを参照してレコード情報を取り出し、ソート配列に格納するためである。検索コアからのレコードソート配列への書き込みは、レコード照合開始時に決まるレコードの管理情報であり、また、検索コアからイベントバッファ配列への書き込みは、照合ヒット情報である。

【0104】

また、読み出しに関して、シグマ検索はソート配列の状態フラグを見るが、これは、シグマ評価（Sigma Evaluation）起動可能判定のためである。シグマ評価は、論理式評価開始に必要なレコード管理情報をソート配列から読み出し、論理式評価のためのヒット情報をイベントバッファから読み出す。

【0105】

シグマバッファフル（Sigma Buffer Full）は、バッファフル時の制御に必要な情報（レコードID、および、*イベントバッファ）をソート配列から読み込む。スケジューラは、検索コアが照合を開始するための情報（*レコード）をソート配列から取り出す。

【0106】

次に、本アクセラレータの主な処理である検索動作（シグマ検索関数が呼ばれてから終了するまで）について述べる。

【0107】

まず、イベントバッファの制御（イベントバッファの読み書きの同期方法）について説明する。

【0108】

図16および図17に示されるように、イベントバッファは、照合結果が格納されるバッファであり、照合ヒット時には検索コアから直接イベントバッファに書き込まれ、論理式評価関数で読み出されて使用される。

【0109】

イベントバッファは、検索コアごとに領域が割り当てられており、その領域を2枚以上のイベントバッファに分割して使用する。なお、領域は有限であるから、いずれバッファをフラッシュして再利用する必要がある。このフラッシュには、シグマバッファフル関数を使用して同期を取るようになっている。

【0110】

ここで、必要な条件判定は、次の通りである。

- A) シグマ評価関数の起動可能条件判定。
- B) シグマバッファフル関数の終了可能条件判定（シグマ評価の再開）。
- C) 検索コアの照合結果を書き込み、未追い越し判定。

なお、イベントバッファを跨いだ情報だけではうまくいかない例としては、照合がイベントバッファを跨いだ情報と、そのレコードの照合が終了した情報の2つがないとうまくいかない。例えば、バッファ距離のみしかなくてそれが2以上だと、次バッファの論理式評価継続可とした場合に、イベントバッファを2つ跨いで終了するようなレコード（イベ

10

20

30

40

50

ントバッファを2枚半程度使うレコード)が照合終了し、その検索コアでは、そのレコードの照合以降はアイドル状態になっているとすると、デッドロックしてしまう。

【0111】

そこで、本実施例の制御方法では、上記の処理を行うときに、前述した状態フラグ(制御データ)とバッファ距離(検索コア内レジスタ、および、I/Oレジスタへのコピー)を使用する。

【0112】

ここで、各変数の操作ルールを説明する。

まず、バッファ距離に関しては、次の通りである。

- ・照合がバッファ終端に到達したら、『1』を加える(+1)
- ・シグマバッファフルから通知を受けたら、『1』を引く(-1)。

10

また、状態フラグに関しては、次の通りである。

- ・初期状態は『0』
- ・照合がイベントバッファを跨いだら『1』とする
- ・照合が終了したら『2』とする

ここで、必要な条件判定方法は、次の通りである。

A) シグマ評価関数の起動可能条件判定。

- ・ソート配列の先頭からレコード状態フラグが『0』以外なら可。

B) シグマバッファフル関数の終了可能条件判定(シグマ評価の再開)。

- ・バッファ距離が『2』以上であれば可(フラッシュ完了前)。
- ・バッファ距離が『1』以上であれば可(フラッシュ完了後)。
- ・レコード状態フラグが『2』であれば可。

20

C) 検索コアの照合結果を書き込み、未追い越し判定。

- ・バッファ距離がイベントバッファ枚数(図16の例では4)未満なら可。

次に、検索動作の2つの例を用いて説明する。なお、その1つの動作は、イベントバッファフラッシュが起きていない通常時の例であり、他の1つは、イベントバッファがフラッシュされる場合の例である。

【0113】

図18A～図18Cはスケジューラの動作を説明するための図であり、検索動作時の時間経過による動作を示すものである。

30

【0114】

図18A～図18Cにおいて、図中の上部にあるブロック(CPU, L2キャッシュ(または、Work RAM), アクセラレータ, メモリ)は、ハードウェアのカテゴリを示し、それらの中にある小さなブロックが動作実体を示す。なお、説明を簡略化するために、図18A～図18Cでは、2つの検索コアSC(検索コア0および1)の構成を示し、また、レコードは、A, B, C, …の順になっているものとする。

【0115】

すなわち、条件は、検索コアが2個、および、レコード順がA, B, C, …で、状況概要は、レコードB照合が終了しても、レコードA照合未終了のため評価待ち、並びに、レコードC照合が終了しても、レコードD照合未終了のため評価待ちとする。

40

【0116】

次に、図18A～図18C中の左側に示す(1)～(6)の各時間範囲における各動作実体の動作について説明する。

【0117】

時間範囲(1)において

・シグマ検索(Sigma Search)：シグマ検索関数が開始すると、メモリにあるXMLデータのレコードインデックスを参照して行き、レコード情報を取り出してソート配列のエントリ個数分を取り込む。登録top, bottomポインタを適切に変更する。なお、この段階では、登録topは1つ目のエントリ、登録bottomは一番下のエントリを指す。そして、登録bottomをスケジューラのマップI/Oレジスタの登録bottom

50

mにコピーする。

【0118】

- ・シグマ評価 (Sigma Evaluation) : 存在していない。
- ・スケジューラ: アイドルモードである。
- ・検索コア: アイドルモードである。

【0119】

時間範囲 (2) において

- ・シグマ検索: スケジューラを検索モードに変更し、論理式評価待ちを開始する。登録 top ポインタが指すソート配列エントリ (レコード A のレコード情報が格納されている) のレコード状態フラグをポーリングする。この時点では、フラグが『0』であるためな

10

【0120】

- ・シグマ評価: 存在していない。
- ・スケジューラ: 検索モードに移行し、照合 top レジスタが示すソート配列エントリから2つのレコード情報を、自回路内のレコード情報バッファ (検索コアの個数分ある) に格納する。照合 top レジスタを『+2』する (ポインタを2エントリ先に進める)。

【0121】

さらに、各検索コアのアイドルフラグを見て、検索コア0および1がアイドル状態であるため、レコード情報 (レコードへのポインタ、および、ソート配列エントリへのポインタ) を渡して照合開始を指示してアイドルフラグをビジー状態にセットする。そして、再度上記と同様にして、レコード情報を自回路内にバッファリングする。

20

【0122】

次に、再度、アイドルフラグレジスタをモニタして、次レコードの分配可能を検索コアと待ち合わせる。この時点ではビジー状態なので、アイドル状態になるのを待つことになる。

【0123】

- ・検索コア: 各検索コア0および1は、スケジューラから渡されたレコードへのポインタから照合を開始する。まず、渡された*ソート配列が指すエントリの検索コア番号と*イベントバッファに対して、それぞれ自分のコア番号と自回路内にある*イベントバッファレジスタの内容を書き込む。さらに、メモリから*レコードから連続する数十バイト (文字バッファのサイズ) のデータ (文字バッファサイズの文字列) を自回路内の文字バッファに取り込む。

30

【0124】

そして、文字バッファから1文字ずつ用いて、自コア内に予め格納されているオートマトンを遷移させる。もし、オートマトンがヒット状態であれば、自コア内の*イベントバッファレジスタが指すイベントバッファエントリにヒット情報を書き込んで、*イベントバッファレジスタを『+1』する。ヒットするごとに同様に処理する。なお、ヒット時の書き込みモードによっては、上書き、或いは、書き込まないなどの動作をすることもある。

【0125】

ここで、文字バッファが空になる前に次の数十バイトを同様に取り込み、検索コアがストールしないようにする。いずれレコードの終端に到着するが、これはレコード文字列を見ることにより検索コアが自分で判別する。

40

【0126】

終端に到着したらスケジューラにアイドル状態を通知し (スケジューラのアイドルフラグにアイドル状態を書きこみ)、同時に、ソート配列のレコード状態フラグを『2』 (レコード照合終了) にする。この時点では、検索コア0および1のどちらもまだ照合終了していないので、レコードの読み込みとオートマトン遷移を続ける。

【0127】

時間範囲 (3) において

50

・シグマ検索：(2)のポーリングを続けている。この時点で、レコードBは照合を終了しているが、先頭レコードAがまだ照合未終了のためポーリングを続ける。

【0128】

・シグマ評価：存在していない。

・スケジューラ：検索コアのアイドルフラグを毎サイクル見ている。暫くすると検索コア1のアイドルフラグが『0（アイドル状態）』となるので、検索コアに3つ目のレコードCのレコード情報を渡して照合を指示する。さらに、(2)と同様に、検索コア1のアイドルフラグを『1（ビジー状態）』にして、1レコード分のレコード情報をソート配列から再度バッファリングする。

【0129】

・検索コア：検索コア0および1は、文字列読み込みとオートマトン遷移を続け、ヒットがあればイベントバッファに書き込む。暫くすると、検索コア1は照合終了するのでスケジューラのアイドルフラグを『0』（アイドル状態）に変更し、スケジューラから渡された*ソート配列が指すエントリのレコード状態フラグに『2』を書き込む。

【0130】

時間範囲 (4) において

・シグマ検索：(2)のポーリングを続けている。この時点でレコードAのレコード状態フラグが『2』となったので、論理式評価が可能。次のエントリのフラグも見て行き、2番目のエントリまでが論理式評価可能なので、レコードAとレコードBを論理式評価するようにシグマ評価を起動する（レコードAのソート配列エントリのアドレスと評価可能レコード数2を引数として渡す）。

【0131】

・シグマ評価：レコードAとレコードBを論理式評価し、ソート配列の*イベントバッファを用いてイベントバッファから照合結果を読んで論理式評価する。レコードID（Record ID）は処理に使用されるが、本アクセラレータには関係しない。この時点では、論理式評価は終了していない。

【0132】

・スケジューラ：(3)と同様の動作となる。ここでは検索コア0のレコードAが照合終了するので、次のレコードDを検索コア0に分配し照合を指示する。(3)と同様に、次レコードの分配を待つ。

【0133】

・検索コア：(3)と同様の動作となる。検索コア0は次レコードDが割り当てられて照合を開始し、また、検索コア1はレコードCの照合を継続する。

【0134】

時間範囲 (5) において

・シグマ検索：なにもしない。

・シグマ評価：論理式評価を継続する。

・スケジューラ：(3)と同様の動作となる。ここでは検索コア0のレコードDが照合終了するので、次のレコードEを検索コア0に分配し照合を指示する。(3)と同様に、次レコードの分配を待つ。

【0135】

・検索コア：(3)と同様の動作となる。検索コア0は次レコードEが割り当てられて照合を開始し、また、検索コア1はレコードCの照合を継続する。

【0136】

時間範囲 (6) において

・シグマ検索：シグマ評価から処理が戻る。ソート配列のレコードAおよびレコードBが格納されていたエントリは不要となったので、(1)と同様にして、XMLデータインデックスを参照して、続きの2つの新しいレコード情報をソート配列に格納する。検索topを『+2』、検索bottomを『+2』する（配列最終要素の次は先頭に循環する）。

。

10

20

30

40

50

【0137】

検索bottomをスケジューラのマップレジスタにコピーする。再度、論理式評価可能待ちでレコードCの状態フラグをポーリングするが、『2』であるので、この時点で直ぐに論理式評価が可能となる。(4)と同様にして、レコードCおよびDを指定して、シグマ評価を起動する。

【0138】

・シグマ評価：レコードAおよびBの論理式評価を継続して終了する。その後、再度呼び出され、レコードCおよびDの論理式評価を開始する。

【0139】

・スケジューラ：(3)と同様の動作となる。ここでは検索コア1のレコードCが照合終了するので、次のレコードFを検索コア1に分配して照合を指示する。(3)と同様に、次レコードの分配を待つ。

【0140】

・検索コア：(3)と同様の動作となる。検索コア1は次レコードFが割り当てられて照合を開始し、また、検索コア0はレコードEの照合を継続する。

【0141】

ここで、図18A～図18Cにおけるシグマ検索の点線で囲まれた部分SSR、並びに、スケジューラの点線で囲まれた部分SDRが繰り返される。このように、照合については、レコード順に関係なく先行するレコードが終了していなくても、検索コアが空くと直ぐに次のレコードが開始される(矢印LB参照)。しかしながら、論理式評価は、レコード順に待ち合わせる(矢印LR参照)必要がある。この制御を、ソート配列で行うようになっている。

【0142】

また、図18A～図18Cに示す実施例では、論理式評価の可能か否かの判定に用いる情報(*登録topの状態フラグ)の待ち合わせにポーリングを用いた場合を説明したが、ハードウェアの割り込みも可能なようにすることもできるのはいうまでもない。

【0143】

図19A～図19Cはイベントバッファ制御の課題を説明するための図であり、イベントバッファが4枚の場合を示している。すなわち、イベントバッファは有限であるため、レコードバッファの照合終了を示すフラグだけでは制御することができない。なお、図19A～図19Cにおいて、参照符号SPは照合結果始点、EPは照合結果終点、WPは照合結果書き込み位置、そして、RPは照合結果読み出し位置を示している。

【0144】

まず、問題がない場合は、図19Aに示されるように、照合結果が1枚のバッファ内に全て収まっている場合である。

【0145】

次に、問題がある第1の場合としては、図19Bに示されるように、照合結果が全てのイベントバッファより大きい場合であり、論理式評価がはじまらず、詰まってしまうことになる。

【0146】

さらに、問題がある第2の場合としては、図19Cに示されるように、書き込みが読み出しを追い抜いてしまう場合であり、照合と論理式評価の同期を取ることが必要となる。

【0147】

図20A～図20Cは各データの書き込みおよび読み出しの関係を説明するための図であり、各バッファ単位で論理式評価を開始してフラッシュを管理する様子を示している。

【0148】

まず、図19Bに示す照合結果が全てノイベントバッファより大きい場合(問題がある第1の場合)には、図20AのPP1に示されるように、照合がイベントバッファの終端に到達した情報を照合情報フラグに付加し、到着時点で論理式評価可能と判定する。

【0149】

次に、図 1 9 C に示す書き込みが読み出しを追い抜いてしまう場合（問題がある第 2 の場合）には、図 2 0 B の P P 2 に示されるように、照合と論理式評価の距離を保持するレジスタを追加し、照合の追い抜きをバッファ単位で防止する。

【0150】

さらに、図 2 0 C の P P 3 に示されるように、論理式評価がイベントバッファ終端に到着したら、そのバッファをフラッシュするようにしてもよい。

【0151】

なお、或る検索コアに対して割り当てられたレコード照合結果（論理式評価）を記録したイベントバッファ（ローカルメモリ：例えば、L 2 キャッシュメモリ）の終端 E P には、その次のイベントバッファ（記憶領域）の先頭へのポインタ（S P '）が記載されている。

10

【0152】

図 2 1 はイベントバッファ制御用情報を説明するための図であり、前述した図 1 5 に対応するものである。

【0153】

図 2 1 に示されるように、論理式評価と照合の同期に使うバッファ距離は、スケジューラ S D におけるバッファ距離レジスタ P Q 1 に格納され、イベントバッファフラッシュの待ち合わせに使うフラッシュ完了フラグは、スケジューラ S D におけるフラッシュ完了フラグ P Q 2 に格納される。

【0154】

20

また、各検索コア S C （検索コア 0 ～ 3）のアイドル状態（空いている状態）は、スケジューラ S D に設けられたレジスタ P Q 3 に保持され、スケジューラ S D が空いている検索コアを判断できるようになっている。

【0155】

なお、共有データにおける複数のレコードを管理する管理情報は、スケジューラ S D に設けられたバッファ P Q 4 に一時的に保持されるようになっている。

【0156】

さらに、照合状態フラグは、レコードの検索コアによる照合状態が記録され、論理式評価可能の判定およびバッファフラッシュ時の論理式評価再開可能の判定に使用されるもので、例えば、L 2 キャッシュの P Q 5 に格納される。なお、検索コア番号は、レコードを照合し、または、照合した検索コアの番号でバッファフラッシュ時に使用されるもので、例えば、L 2 キャッシュの P Q 6 に格納される。

30

【0157】

図 2 2 A ～ 図 2 2 C はイベントバッファフラッシュ時の動作例を説明するための図であり、イベントバッファフラッシュ時の時間経過による動作を示すものである。

【0158】

図 1 8 A ～ 図 1 8 C を参照して述べた例では、イベントバッファサイズが無限にあるケースを想定して説明した。しかしながら、実際には、イベントバッファサイズは有限であるため、イベントバッファを 2 枚以上に分割して切り替えながら使うことになる。ここでは、その場合の例を説明する。

40

【0159】

なお、説明を簡略化するために、図 2 2 A ～ 図 2 2 C では、検索コアが 1 個でイベントバッファが 2 枚の構成を示し、また、あるレコード A の照合結果のサイズが合計 2 枚以上のイベントバッファサイズとなる場合を示す。

【0160】

すなわち、条件は、検索コアが 1 個、および、イベントバッファが 2 枚で、状況概要は、レコード A の照合がバッファ跨いだので即評価可能であり、1 度目のバッファフラッシュ時は、レコード A の照合が 2 つ目のバッファ跨いだので即終了可能であり、そして、2 度目バッファフラッシュ時は、レコード A 照合が 3 つ目のバッファ跨いでいないので待ち合わせとする。

50

【0161】

次に、図22A～図22C中の左側に示す(1)～(7)の各時間範囲における各動作実体の動作について説明する。

【0162】

時間範囲 (1) において

・シグマ検索：レコードAの論理式評価可能を、レコードAの状態フラグをポーリングして待っている。この段階では、ポーリング中である。

【0163】

・シグマ評価：存在していない。
・シグマバッファフル：存在していない。
・スケジューラ：次レコードBの分配可能を、検索コアのアイドル (idle) フラグを見ることが待つ。この段階では、『1』（ビジー状態）を示したままであるため待ち続けることになる。

10

【0164】

・検索コア：レコードAの照合中であり、通常時と同様に、メモリからレコード文字列を持ってきてオートマトンを遷移する。そして、ヒットしたらイベントバッファにヒット情報を書き込む。ヒット情報を書き込んでいるうちに、書き込み先のアドレスが1枚目のイベントバッファの終端に到着したら、ソート配列の状態フラグに『1』を書き込む（イベントバッファ跨ぎを通知する）。そして、バッファ距離レジスタを『+1』する。

【0165】

20

時間範囲 (2) において

・シグマ検索：状態フラグをポーリングすると、状態フラグが『1』となっているので、レコードAを論理式評価するように、シグマ評価を呼び出す。

【0166】

・シグマ評価：渡されたレコードAを、通常時と同様に、論理式評価を開始する。このとき、イベントバッファからヒット情報を参照する。

【0167】

・シグマバッファフル：存在していない。
・スケジューラ：(1)と同じで、変化なし。
・検索コア：2枚目のイベントバッファにヒット情報を書き出す。他は、(1)と同じである。

30

【0168】

時間範囲 (3) において

・シグマ検索：シグマ評価の終了を待つ。
・シグマ評価：(2)の論理式評価を継続中である。
・シグマバッファフル：存在していない。
・スケジューラ：(2)と同じ。変化なし。

【0169】

・検索コア：(1)と同様にヒット情報を書き込んでいるうちに、書き込み先のアドレスが1枚目のイベントバッファの終端に到着。今回は既に状態フラグを『1』にしてあるので、特に外部に通知はしない。バッファ距離レジスタを『+1』する。このときバッファ距離レジスタ=2で、イベントバッファ枚数（2枚）未満でないので、1枚目のイベントバッファが論理式評価で消費されて終わっていないためストールする。

40

【0170】

時間範囲 (4) において

・シグマ検索：(3)と同じくシグマ評価の終了を待つ。
・シグマ評価：論理式評価中に参照するイベントバッファのヒット情報の内容がバッファ終端を示している（イベントバッファの終端には、終端を示すビットパターンと次イベントバッファへのポインタが予め書き込まれている）ため、シグマバッファフルを呼び出す。引数にはこのレコードのソート配列エントリへのポインタを渡す。しばらくすると、

50

シグマバッファフルが終了し、次イベントバッファの先頭から論理式評価を再開する。

【0171】

・シグマバッファフル：渡されたソート配列エントリへのポインタから、検索コア番号を割り出す。そして対応するフラッシュ完了フラグが『0』であるか否かを確認する。『0』なのでフラグに『1』を書き込み、バッファ距離コピーを獲得しておく。

【0172】

また、検索コア番号をスケジューラに渡し、その検索コアのイベントバッファフラッシュを指示する。（この例では、検索コアが1つしかないため検索コア番号を指定する必要はないが、一般のケースでは必要となる。）

【0173】

次に、先に獲得しておいたバッファ距離コピー（IOレジスタ）が2以上、または、状態フラグが『2』であるか否かを判定する。バッファ距離コピーが『2』なのですぐに関数が終了可能である。論理式評価再開可能の待ち合わせを行う。

【0174】

・スケジューラ：フラッシュ指示と検索コア番号0が与えられたので、対応する検索コア0にフラッシュ指示する。

【0175】

・検索コア：ストール中にフラッシュ指示命令が与えられたので、バッファ距離レジスタを『-1』し、その内容をスケジューラのバッファ距離レジスタのコピーに上書きする。そして、フラッシュ完了フラグを『0』にする。さらに、バッファ距離が『1』となったので照合を再開する。

【0176】

時間範囲 (5) において

・シグマ検索：(4)と同じくシグマバッファフルの終了を待つ。

・シグマ評価：(4)と同様にイベントバッファ終端に到着したので、再度シグマバッファフルを呼び出す。その後は、シグマバッファフルの終了を待つ。

【0177】

・シグマバッファフル：スケジューラへの指示までは(4)と同じ。もしこの時点でフラッシュ完了フラグが『1』なら待ち合わせる（1回前のフラッシュ。この場合は関係ない）。次に、予め保持しておいたバッファ距離が『1』なので、フラッシュ完了フラグとバッファ距離をポーリングして待ち合わせ。（フラッシュ完了フラグ=0かつ（バッファ距離 $\geq$ 1または状態フラグ=2））となるのを待つ。

【0178】

・スケジューラ：(4)と同様にフラッシュを指示する。

・検索コア：(4)と同様に、フラッシュし、照合を継続する。

【0179】

時間範囲 (6) において

・シグマ検索：(5)と同じく、シグマ評価の終了を待つ。

・シグマ評価：シグマバッファフルの終了を待つ。しばらくすると、シグマバッファフルが終了し、次イベントバッファの先頭から論理式評価を再開する。

【0180】

・シグマバッファフル：(5)に引き続きポーリングを行い、しばらくすると、バッファ距離=1であるが、状態フラグ=2となるので終了する。

【0181】

・スケジューラ：次レコードの分配を待ち、検索コア0からアイドル状態が通知されるので次のレコードBを分配する。レコード情報バッファを再度充填して、また次レコードの分配待ち状態になる。

【0182】

・検索コア：レコード照合が終了し、それをソート配列とスケジューラに通知する。次のレコードBが渡されて、照合を開始する。

10

20

30

40

50

【0183】

時間範囲 (7) において

・シグマ検索：(6)に引き続き、シグマ評価の終了を待つ。しばらくすると終了するので、空いたソート配列のエントリに続きのレコード情報を充填し、論理式評価可能を待つ。この時点では、レコードBが照合終了していないので待つことになる。

【0184】

- ・シグマ評価：論理式評価が終了し、関数が終了する。
- ・シグマバッファフル：存在していない。
- ・スケジューラ：(6)に引き続き次レコードの分配を待つ。
- ・検索コア：(6)に引き続きレコードBを照合する。

10

【0185】

以上、実施例における動作を説明してきたが、次に、各関数の検索動作を、フローチャートを参照して説明する。

【0186】

図23はAPI関数と呼び出し手続きをフローチャートと関連付けて説明するための図である。図23において、左側部の図は、図5に示す検索シーケンスを書き換えたものに相当し、また、中央の図は、各API関数と呼び出し手続きを説明するために使用した図13と同様のものである。

【0187】

図23に示されるように、本実施例が対象とする動作は、検索シーケンスにおけるレコード照合処理、照合結果処理、論理式評価処理および検索結果処理に対応する検索動作である。ここで、検索動作（図23中で太線で囲んだ部分）において、照合実行起動135は、図24に示すシグマ検索処理に対応し、論理式評価137は、図27に示すシグマ評価処理に対応し、そして、アクセラレータライブラリ138内のバッファフル処理139は、図28に示すシグマバッファフル処理に対応する。

20

【0188】

以下、各フローチャートを参照して、シグマ検索処理、シグマ評価処理およびシグマバッファフル処理を説明する。

【0189】

図24はシグマ検索処理の一例を示すフローチャートである。ここで、シグマ検索（Sigma Search）処理は、メインルーチンであり、検索動作をはじめめるための処理（関数）である。

30

【0190】

シグマ検索処理を介しすると、まず、ステップST1において、はじめにXMLインデックスを参照して（なめて）、ソート配列にレコード情報を格納し、ステップST2に進む。

【0191】

ステップST2では、アクセラレータを照合モードに変更し、ステップST3に進んで、ポーリングを行う。ここで、ポーリングでは、次評価レコードエントリの状態フラグをポーリングする。

40

【0192】

さらに、論理式評価可能となったら、ステップST4において、シグマ評価（Sigma Evaluation）の引数を決定し、ステップST5に進んでシグマ評価を呼ぶ。ここで、渡すレコードの個数に下限や上限を設けてもよい。

【0193】

次に、ステップST6に進んで、シグマ評価の終了を待ち、さらに、ステップST7に進んで、シグマ評価が終了したら、渡したレコード情報が格納されているソート配列のエントリに新しいレコード情報を充填してステップST8に進む。

【0194】

ステップST8では、XMLデータのXMLインデックスが終端になったかどうかを判

50

定する。ステップ S T 8 において、XML データの XML インデックスが終端になっていないと判定されると、ステップ S T 3 に戻って同様の処理を繰り返す。

【0195】

一方、ステップ S T 8 において、XML データの XML インデックスが終端になったと判定されると、ステップ S T 9 に進んで、アクセラレータにアイドル状態への移行条件を通知してステップ S T 10 に進む。

【0196】

ステップ S T 10 では、論理式評価が終了したかどうかを判定し、論理式評価が終了したと判定すると関数を終了し、そうでなければステップ S T 3 のポーリングに戻り、同様の処理を繰り返す。

10

【0197】

図 25 は図 24 のシグマ評価を 1 エントリごとに待ち合わせる場合を示すフローチャートであり、図 26 は図 24 のシグマ評価を複数エントリごとに待ち合わせる場合を示すフローチャートである。

【0198】

図 25 に示されるように、図 24 のシグマ評価を 1 エントリごとに待ち合わせる場合、まず、ステップ S T 21 において、ソート配列の全エントリに対して、レコード順（レコードの順番付けの順）にレコード管理情報を格納し、次に、ステップ S T 22 に進んで、アクセラレータを起動する。

【0199】

さらに、ステップ S T 23 に進んで、ソート配列の先頭エントリの照合完了を待ち合わせて、ステップ S T 24 に進む。ステップ S T 24 では、先頭エントリのレコードの論理式評価を実行して、ステップ S T 25 に進む。

20

【0200】

ステップ S T 25 では、ソート配列に 1 レコード分のレコード管理情報を充填し、先頭エントリを 1 つ進める。

【0201】

そして、ステップ S T 26 に進んで、データベースの全レコード処理が終了したかどうかを判定し、終了していないと判定されれば、ステップ S T 23 に戻って同様の処理を繰り返し、データベースの全レコード処理が終了したと判定されると処理を終了する。

30

【0202】

図 26 に示されるように、図 24 のシグマ評価を複数エントリごとに待ち合わせる場合、まず、ステップ S T 31 において、ソート配列の全エントリに対して、レコード順にレコード管理情報を格納し、次に、ステップ S T 32 に進んでアクセラレータを起動する。

【0203】

さらに、ステップ S T 33 に進んで、ソート配列の先頭エントリの照合完了を待ち合わせて、ステップ S T 34 に進む。ステップ S T 34 では、先頭エントリからレコード順に連続して照合が終了した複数のレコードの論理式評価をレコード順に実行して、ステップ S T 35 に進む。

【0204】

ステップ S T 35 では、ソート配列に論理式評価を行ったレコード数と同じ数だけレコード管理情報を充填し、先頭エントリを処理した数だけ進める。

40

【0205】

そして、ステップ S T 36 に進んで、データベースの全レコード処理が終了したかどうかを判定し、終了していないと判定されれば、ステップ S T 33 に戻って同様の処理を繰り返し、データベースの全レコード処理が終了したと判定されると処理を終了する。

【0206】

図 27 はシグマ評価処理の一例を示すフローチャートである。ここで、シグマ評価 (Sigma Evaluation) は、論理式評価を行うための関数であり、シグマ検索で論理式評価可能と判断されたら呼び出される。

50

【0207】

図27に示されるように、依頼された先頭レコードのソート配列エントリアドレスとそこからのレコード個数が渡されると、まず、ステップST41において、 $a = (\text{レコードソート構造体配列の現在のレコードのイベントバッファのアドレス})$ 、 $b = 1$ として、ステップST42に進む。ステップST42において、ソート配列からレコードIDを取得し、通常動作のステップST5（通常動作のループ）に進む。すなわち、先頭レコードを現在のレコードとして論理式評価を開始し、イベントバッファから各エントリを読んで評価する。

【0208】

ステップST5は、ステップST51～ST54で構成され、ま、ステップST51において、 a が示す先の照合結果を読んで論理式評価を実行する。

10

【0209】

次に、ステップST52に進んで、 $a++$ とし、さらに、ステップST53に進んで、 $(a \text{ が示す先のエントリ}) == (\text{レコード結果の終端エントリ})$ かどうかを判定する。

【0210】

ステップST53において、 $(a \text{ が示す先のエントリ}) == (\text{レコード結果の終端エントリ})$ であると判定されると、レコード終了時ループ（ステップST6）のステップST61に進み、逆に、 $(a \text{ が示す先のエントリ}) == (\text{レコード結果の終端エントリ})$ ではないと判定されると、ステップST54に進む。

20

【0211】

ステップST61では、 $b == (\text{渡されたレコード個数})$ かどうかを判定し、 $b == (\text{渡されたレコード個数})$ ではないと判定されると、ステップST62で、 $a = (\text{次レコード照合結果の先頭アドレス})$ 、 $b++$ として、ステップST42に戻り同様の処理を繰り返す。そして、ステップST61において、 $b == (\text{渡されたレコード個数})$ であると判定されると処理を終了する。

【0212】

すなわち、もし、現在のレコードが終了したら、次レコードをセットして通常動作に戻り、現在のレコードが終了しなければ終了する。

【0213】

ステップST54では、 $(a \text{ が示す先のエントリ}) == (\text{イベントバッファ終端のエントリ})$ かどうかを判定し、 $(a \text{ が示す先のエントリ}) == (\text{イベントバッファ終端のエントリ})$ であると判定されると、バッファフル時のステップST7におけるステップST71に進んで、シグマバッファフルを起動して、ステップST72に進む。

30

【0214】

そして、ステップST72において、 $a = (\text{次イベントバッファの先頭エントリのアドレス})$ としてステップST51に戻り、また、ステップST54で、 $*a == (\text{イベントバッファ終端のエントリではないと判定された場合もステップST51に戻り、同様の処理を繰り返す。})$

【0215】

すなわち、イベントバッファ終端に到達したらシグマバッファフルを呼び出して帰ってくるのを待ち、また、通常動作に戻る。

40

【0216】

図28シグマバッファフル処理の一例を示すフローチャートである。ここで、シグマバッファフル (Sigma Buffer Full) は、イベントバッファの切り替わりを司る関数であり、対応する検索コアのイベントバッファのフラッシュを指示し、そして、シグマ評価でイベントバッファ終端に達したら呼び出される。

【0217】

図28に示されるように、まず、ステップST81において、渡されたソート配列要素に対応する検索コア番号を検索し、ステップST82に進む。すなわち、フラッシュするイベントバッファのソート配列エントリのアドレスが引数として渡されるが、そのアドレ

50

スからソート配列を参照して検索コア番号を獲得する。

【0218】

ステップST82では、フラッシュ完了フラグ=1かどうかを判定し、フラッシュ完了フラグ=1ではないと判定されるまで処理を続け、フラッシュ完了フラグ=1ではないと判定されると、ステップST83に進む。

【0219】

ステップST83において、フラッシュ完了フラグ=1として、ステップST84に進み、バッファ距離コピーを獲得する。すなわち、その検索コアに対する前回のフラッシュ指示との待ち合わせを行い、フラッシュ指示が可能であればその時点でスケジューラのI/Oレジスタであるバッファ距離コピーを獲得しておく。

10

【0220】

さらに、ステップST85に進んで、スケジューラにバッファフラッシュ指示と検索コア番号の通知を行って、ステップST86に進む。

【0221】

ステップST86では、バッファ距離コピー ≥ 2 、または、状態フラグ=2かどうかを判定し、バッファ距離コピー ≥ 2 ではなく、且つ、状態フラグ=2ではないと判定されると、ポーリングを行うステップST9に進み、バッファ距離コピー ≥ 2 、または、状態フラグ=2であると判定されると、処理を終了する。

【0222】

すなわち、先に獲得しておいたバッファ距離コピーの値とソート配列の状態フラグから終了可能かどうかを判定する。なお、終了すると論理式評価が再開される。

20

【0223】

そして、ステップST9においてはステップST91～ST93において、フラッシュ完了フラグをポーリングし、次に状態フラグとバッファ距離コピーをポーリングして可能状態を待ち合わせる。

【0224】

すなわち、ステップST91では、フラッシュ完了フラグ=0かどうかを判定し、フラッシュ完了フラグ=0であると判定されるまで処理を続け、フラッシュ完了フラグ=0であると判定されると、ステップST92に進み、バッファ距離コピーを再度獲得して、ステップST93に進む。

30

【0225】

ステップST93では、バッファ距離 ≥ 1 、または、状態フラグ=2であるかどうかを判定し、バッファ距離コピー ≥ 2 ではなく、且つ、状態フラグ=2ではないと判定されると、ステップST92に戻って、同様の処理を繰り返し、バッファ距離 ≥ 1 、または、状態フラグ=2であると判定されると、処理を終了する。

【0226】

上述した各実施例では、主としてオートマトン処理を行う検索装置および検索方法を説明したが、一般的な並列処理システムであって、その入力データおよび中間データに特徴があり、その特徴のあるデータの処理に対して所定のルールに従って処理する検索装置および検索方法に対しても適用することが可能である。

40

【0227】

以上の実施例を含む実施形態に関し、さらに、以下の付記を開示する。

(付記1)

CPUと、複数の検索コアおよびスケジューラを有するアクセラレータと、を備えた検索装置であって、

前記CPUは、入力された検索式に従ってオートマトンを作成し、

前記スケジューラは、処理対象テキストを、順番付けしたレコード単位で前記各検索コアに分配し、

前記各検索コアは、前記分配された各レコードをそれぞれ並列的にオートマトン照合し、

50

前記CPUは、前記各レコード単位の照合結果を該各レコードの前記順番付けの順で論理式評価することを特徴とする検索装置。

【0228】

(付記2)

付記1に記載の検索装置において、

前記処理対象テキストの前記各レコード、および、前記各レコードの前記照合結果は、それぞれのデータサイズに規則性を持たないことを特徴とする検索装置。

【0229】

(付記3)

付記1または2に記載の検索装置において、さらに、

前記CPUおよび前記検索コアの共有データを格納するローカルメモリを備え、
該共有データは、前記複数のレコードの照合結果を格納する照合結果情報と、複数のレコードを管理する管理情報とを有し、

前記各レコードの前記順番付けは、前記管理情報として前記ローカルメモリに保持されることを特徴とする検索装置。

【0230】

(付記4)

付記3に記載の検索装置において、

前記照合結果情報は、前記各検索コアに対して前記ローカルメモリの2枚以上の記憶領域に分割して割り当てて管理されることを特徴とする検索装置。

【0231】

(付記5)

付記4に記載の検索装置において、

前記各検索コアに対して割り当てられた前記記憶領域の終端には、次の記憶領域の先頭へのポインタが記載されていることを特徴とする検索装置。

【0232】

(付記6)

付記3に記載の検索装置において、

前記管理情報は、配列またはリスト構造により照合すべきテキスト群の順番を保持することを特徴とする検索装置。

【0233】

(付記7)

付記3～6のいずれか1項に記載の検索装置において、

前記スケジューラは、前記管理情報を読んで空いている前記検索コアに前記順番付けの順で前記管理データを分配して照合を指示し、検索コアはランダムに照合完了することを特徴とする検索装置。

【0234】

(付記8)

付記7に記載の検索装置において、

前記スケジューラは、前記空いている前記検索コアを判断するための情報を保持するレジスタを備えることを特徴とする検索装置。

【0235】

(付記9)

付記7または8に記載の検索装置において、

前記スケジューラは、前記管理情報を一時的に保持するバッファを備えることを特徴とする検索装置。

【0236】

(付記10)

付記1～9のいずれか1項に記載の検索装置において、

前記処理対象テキストは、構造化文書であり、

前記各検索コアは、前記構造化文書のデータから所定の文字列を検索することを特徴とする検索装置。

【0237】

(付記11)

付記10に記載の検索装置において、

前記構造化文書は、XML文書であることを特徴とする検索装置。

【0238】

(付記12)

付記1～11のいずれか1項に記載の検索装置を複数結合したことを特徴とする半導体チップ。

10

【0239】

(付記13)

付記12に記載の半導体チップにおいて、該半導体チップは付記1～8のいずれか1項に記載の検索装置を2個備えたことを特徴とする半導体チップ。

【0240】

(付記14)

付記12または13に記載の半導体チップを複数結合し、

該各半導体チップに対してそれぞれ設けた複数のメモリを備えることを特徴とするプロセッサモジュール。

【0241】

20

(付記15)

付記14に記載のプロセッサモジュールを複数結合すると共に、前記各検索コアがアクセス可能な主記憶装置を備えることを特徴とする検索サーバ。

【0242】

(付記16)

付記15に記載の検索サーバと、

検索対象のデータが格納されたストレージサーバと、

前記検索サーバおよび前記ストレージサーバを管理するディレクタサーバと、を備える検索システム。

【0243】

30

(付記17)

CPUと、複数の検索コアおよびスケジューラを有するアクセラレータと、を備えた検索装置を使用した検索方法であって、

前記CPUが、入力された検索式に従ってオートマトンを作成し、

前記スケジューラが、処理対象テキストを順番付けされたレコード単位で前記各検索コアに分配し、

前記各検索コアが、前記分配された各レコードをそれぞれ並列的に照合処理し、

前記CPUが、前記各レコード単位の照合結果を該各レコードの前記順番付けの順で論理式評価することを特徴とする検索方法。

【0244】

40

(付記18)

付記17に記載の検索方法において、

前記処理対象テキストの前記各レコード、および、前記各レコードの前記照合結果は、それぞれのデータサイズに規則性を持たないことを特徴とする検索方法。

【0245】

(付記19)

付記17または18に記載の検索方法において、

前記スケジューラは、前記CPUおよび前記検索コアの共有データにおける複数のレコードを管理する管理情報を読んで空いている前記検索コアに前記順番付けの順で前記管理データを分配して照合を指示し、検索コアはランダムに照合完了することを特徴とする検

50

索方法。

【0246】

(付記20)

CPUと、複数の検索コアおよびスケジューラを有するアクセラレータと、を備えたコンピュータに、

前記CPUに、入力された検索式に従ってオートマトンを作成させる手順と、

前記スケジューラに、処理対象テキストを順番付けされたレコード単位で前記各検索コアに分配させる手順と、

前記各検索コアに、前記分配された各レコードをそれぞれ並列的に照合処理させる手順と、

10

前記CPUに、前記各レコード単位の照合結果を該各レコードの前記順番付けの順で論理式評価させる手順と、を実行させる検索プログラム。

【図面の簡単な説明】

【0247】

【図1A】オートマトンによるキーワード検索アルゴリズムの一例であるシグマアルゴリズムを説明する図(その1)である。

【図1B】オートマトンによるキーワード検索アルゴリズムの一例であるシグマアルゴリズムを説明する図(その2)である。

【図2】検索装置の一例を概略的に説明するための図である。

【図3】従来の検索装置における検索シーケンスの一例を概略的に示す図である。

20

【図4】従来の検索方法による処理順を概略的に示す図である。

【図5】本実施形態の検索装置における検索シーケンスの一例を概略的に示す図である。

【図6】本実施形態の検索方法による処理順を概略的に示す図である。

【図7】検索装置の一実施例の要部を示すブロック図である。

【図8】検索装置の他の実施例の要部を示すブロック図である。

【図9】各実施例が適用される検索サーバの一例を概略的に示す図である。

【図10】図9の検索サーバを含む検索システムの全体構成を概略的に示す図である。

【図11】レコードIDからレコードを特定する処理を説明するための図である。

【図12】各モジュールの役割を説明するための図である。

【図13】各API関数と呼び出し手続きを説明するための図である。

30

【図14】制御データの詳細なデータ構造を示す図である。

【図15】共有データの構造を説明するための図である。

【図16】各データの書き込みおよび読み出しの関係を説明するための図(その1)である。

【図17】各データの書き込みおよび読み出しの関係を説明するための図(その2)である。

【図18A】スケジューラの動作を説明するための図(その1)である。

【図18B】スケジューラの動作を説明するための図(その2)である。

【図18C】スケジューラの動作を説明するための図(その3)である。

【図19A】イベントバッファ制御の課題を説明するための図(その1)である。

40

【図19B】イベントバッファ制御の課題を説明するための図(その2)である。

【図19C】イベントバッファ制御の課題を説明するための図(その3)である。

【図20A】各データの書き込みおよび読み出しの関係を説明するための図(その1)である。

【図20B】各データの書き込みおよび読み出しの関係を説明するための図(その2)である。

【図20C】各データの書き込みおよび読み出しの関係を説明するための図(その3)である。

【図21】イベントバッファ制御用情報を説明するための図である。

【図22A】イベントバッファフラッシュ時の動作例を説明するための図(その1)であ

50

る。

【図 2 2 B】 イベントバッファフラッシュ時の動作例を説明するための図（その 2）である。

【図 2 2 C】 イベントバッファフラッシュ時の動作例を説明するための図（その 3）である。

【図 2 3】 A P I 関数と呼び出し手続きをフローチャートと関連付けて説明するための図である。

【図 2 4】 シグマ検索処理の一例を示すフローチャートである。

【図 2 5】 図 2 4 のシグマ評価を 1 エントリごとに待ち合わせる場合を示すフローチャートである。

10

【図 2 6】 図 2 4 のシグマ評価を複数エントリごとに待ち合わせる場合を示すフローチャートである。

【図 2 7】 シグマ評価処理の一例を示すフローチャートである。

【図 2 8】 シグマバッファフル処理の一例を示すフローチャートである。

【符号の説明】

【 0 2 4 8 】

1 1 検索対象文書

1 2 入力文字

1 3 現在の状態

1 4 入力アドレス

20

1 5 メモリ

1 6 出力データ

1 7 次の状態

A C アクセラレータ

C H I P 半導体チップ

D S ディレクタサーバ

M E M メモリコントローラ

N E T ネットワーク

P B プロセッサボックス

P M プロセッサモジュール

30

P S プロセッサシステム（検索装置）

R S ラックシステム

S C 検索コア（照合装置）

S D スケジューラ

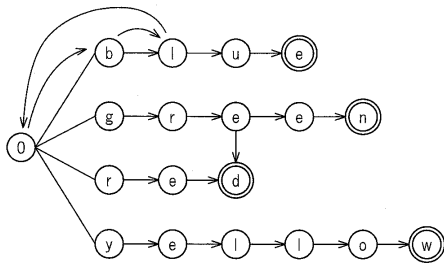
S S サーチャサーバ

S T R ストレージ

【図 1 A】

Fig.1A

[検索キーワードオートマトン]



【図 1 B】

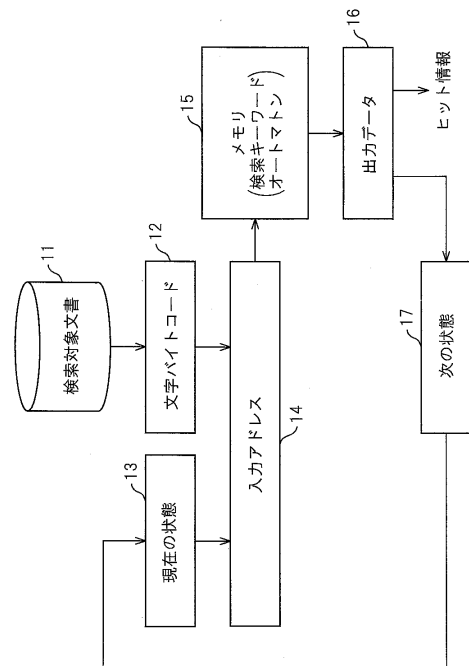
Fig.1B

[文書]

... black ...

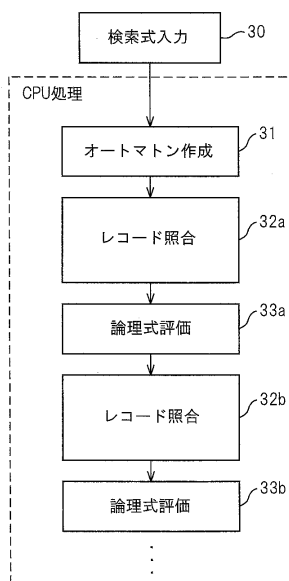
【図 2】

Fig.2



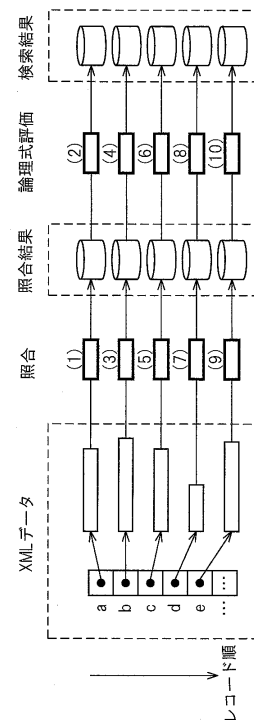
【図 3】

Fig.3



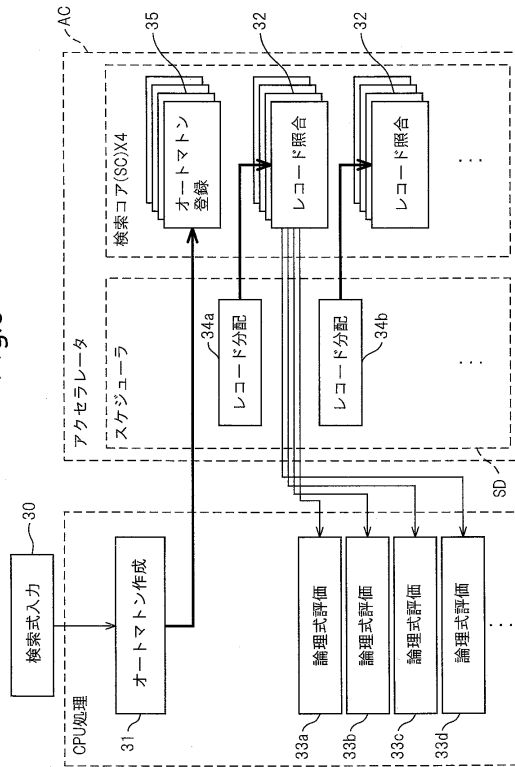
【図 4】

Fig.4



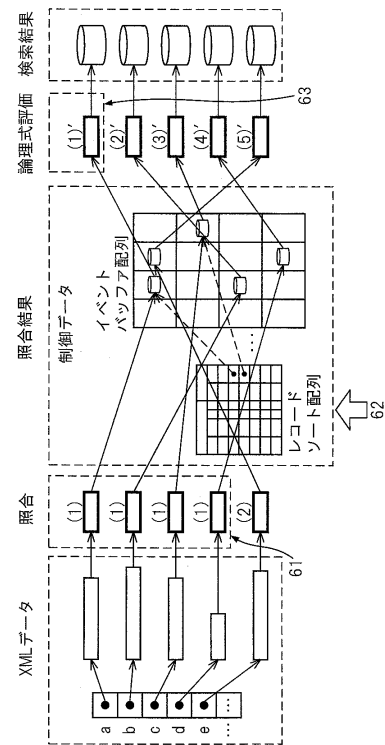
【図 5】

Fig.5



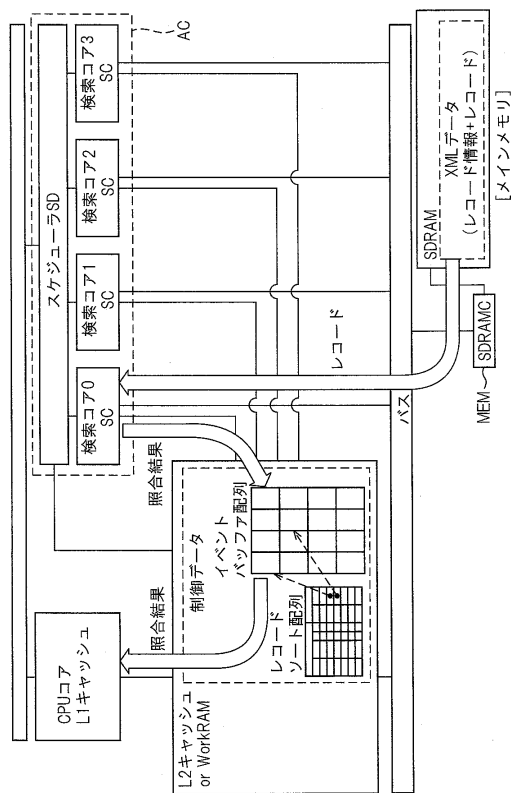
【図 6】

Fig.6



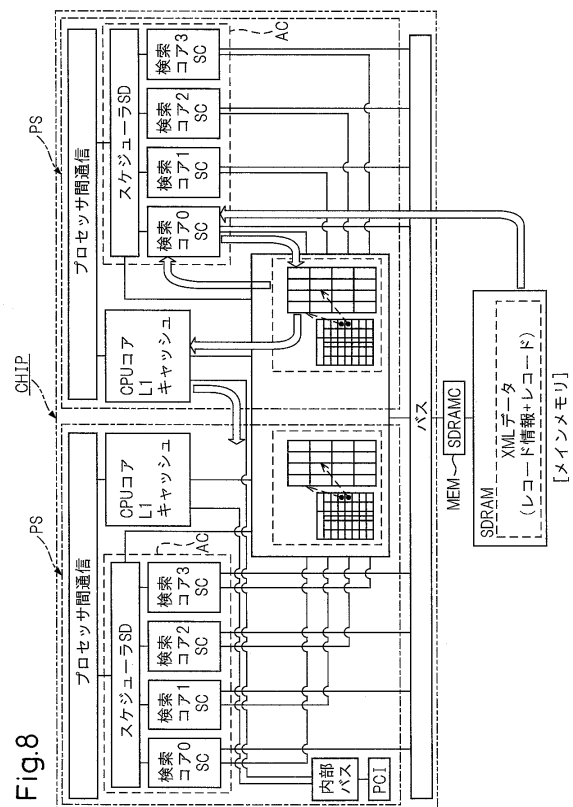
【図 7】

Fig.7



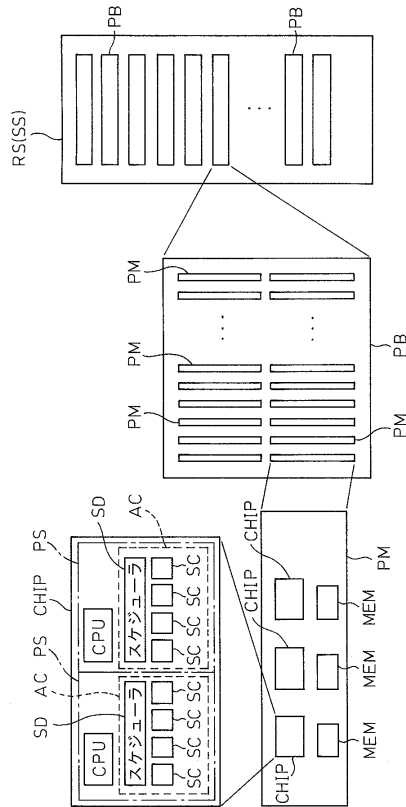
【図 8】

Fig.8



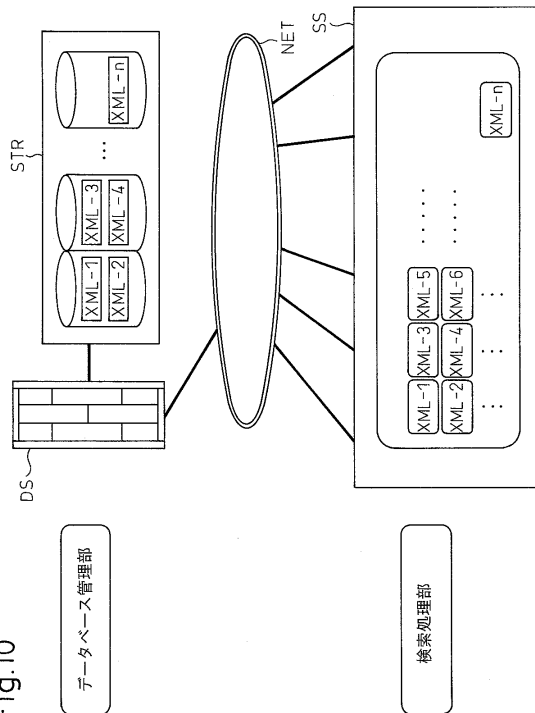
【図 9】

Fig.9



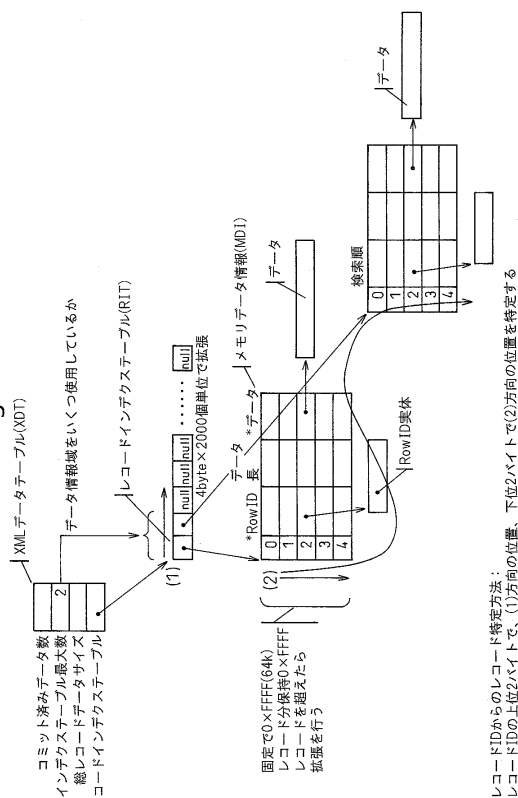
【図 10】

Fig.10



【図 11】

Fig.11

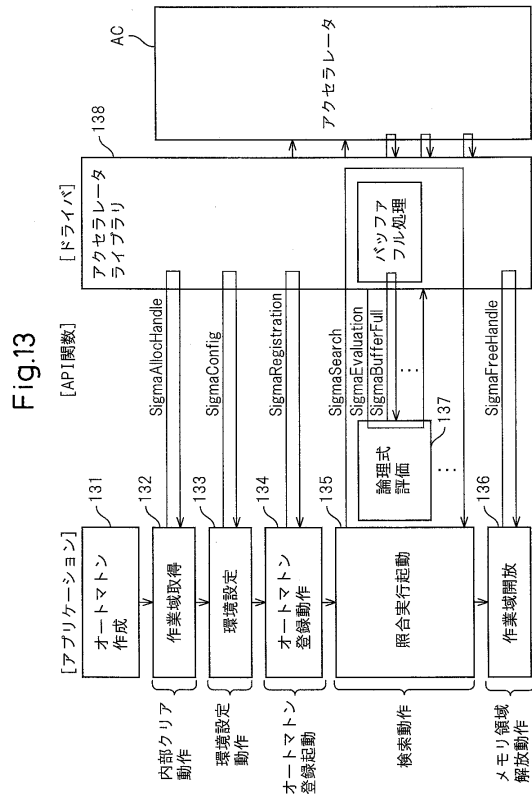


【図 12】

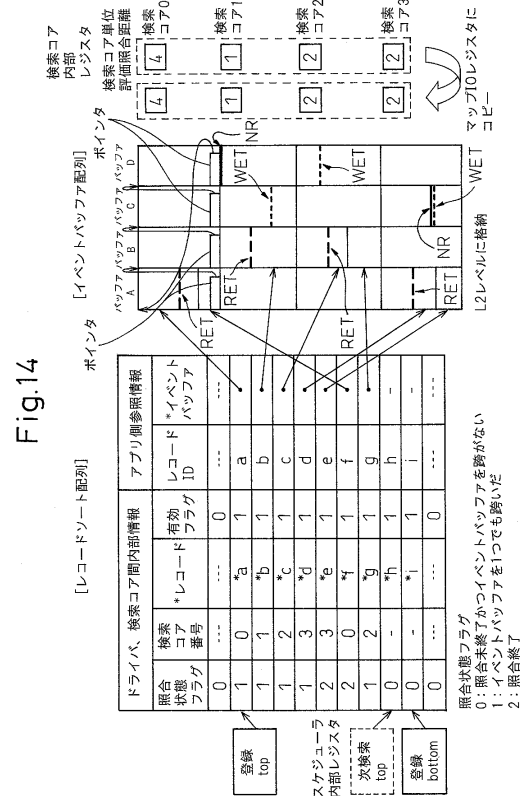
Fig.12

大分類	モジュール		役割
	中分類	小分類	
ソフト	アプリ	シグマ評価 (SigmaEvaluation)	論理式評価を行う。 バッファフルを抽出したらシグマバッファフルを呼ぶ。
	ドライバ	シグマ検索 (SigmaSearch)	XMLのインデックスをなめて、レコードへのポインタを インオーダーで取り出し、ソート配列に登録する。適当な タイミング(ポーリング)でバッファフル状態フラグをみて、 論理式評価関数を起動する。論理式評価関数が終了したら、 そのまでのソート配列エントリをフラッシュ。 適当なタイミング(論理式評価関数が終了したら評価終了 レコード種数をカウント、閾値を設けて起動などで) レコード情報をソート配列に補充する。
		シグマバッファ フル (SigmaBufferFull)	論理式評価がイベントバッファ終端に到達したら、呼ばれる。 イベントバッファフラッシュを行い、現在論理式評価中の レコードの照会結果が終了または次イベントバッファが フルになるまで待ち、終了する。
ハード	スケジューラ	-	アイドル(idle)な検索コアがあったら、ソート配列に 登録された照会未開始の先頭のレコード情報 (ポインタなど)を検索コアに分配し、検索開始を指示する
	検索コア	-	レコード情報が与えられたら、メモリからレコードの文字列を 読み込む照会を行う。ヒットしたらイベントバッファに書き 込み、同時に、照会が終了する、イベントバッファを腾空 などがあった場合は、バッファフル状態フラグを通知し変更。照会 終了したら、スケジューラにアイドル状態を通知し停止する。 イベントバッファフルなどがあつたらストローし、 イベントバッファフルがフラッシュされるのを待つ。

【図 13】



【図 14】



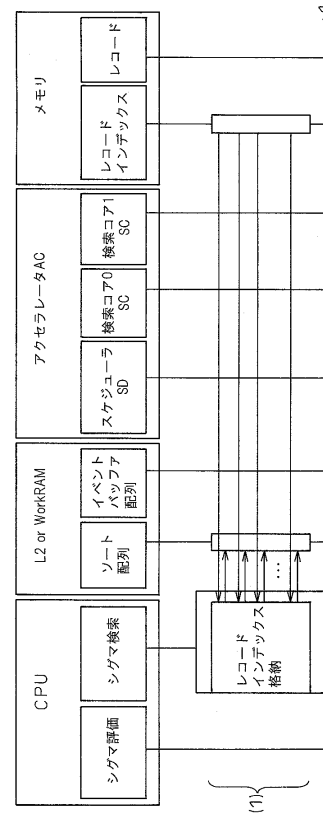
【図 17】

Fig.17

大項目	小項目	CPU						アクセラレータAC	
		シグマバッファフル		シグマ検索		シグマ評価		スケジュールSD	検索コアSC
		書き	読み	書き	読み	書き	読み	書き	読み
レコード ソート配列	状態フラグ	x	○	x	○	x	x	x	x
	検索コア番号	x	○	x	x	x	x	○	x
	*レコード	x	x	○	x	x	x	x	x
	有効フラグ	x	x	○	○	x	○	x	x
	レコードID	x	x	○	x	x	○	x	x
イベント バッファ配列	*イベント バッファ	x	x	x	x	x	○	x	○
	-	x	x	x	x	x	○	x	○
	バッファ距離 (コピー)	x	○	x	x	x	x	x	x
	I/Oレジスタ	x	x	x	x	x	x	x	x

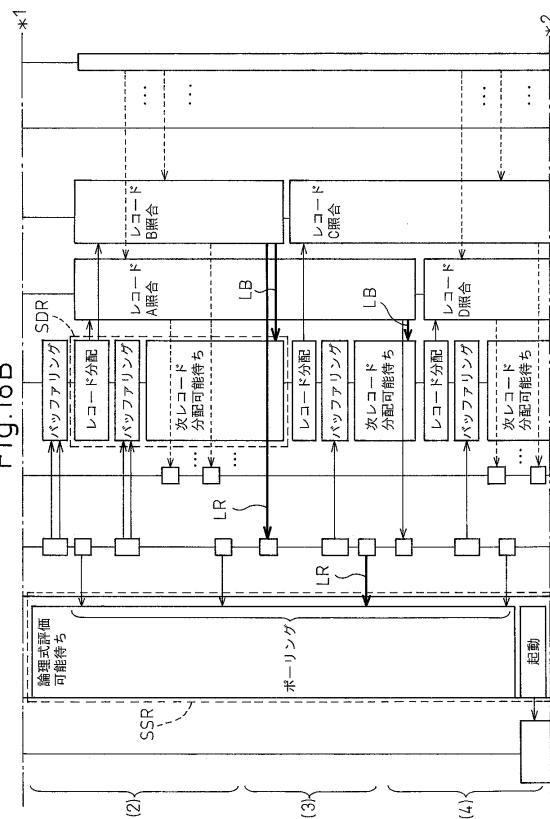
【図 18 A】

Fig.18A



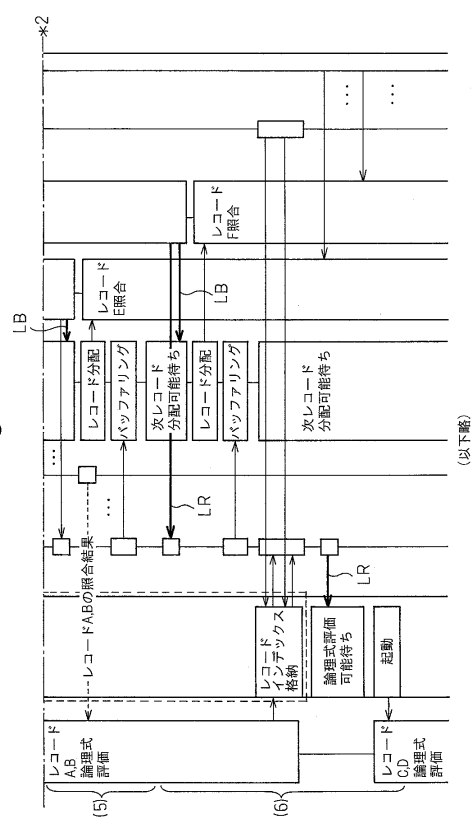
【図 18 B】

Fig.18B

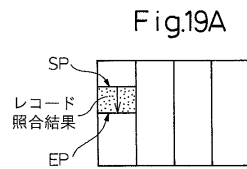


【図 18 C】

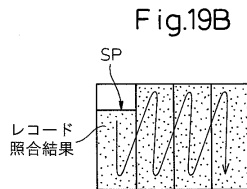
Fig.18C



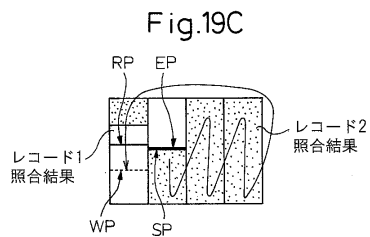
【図19A】



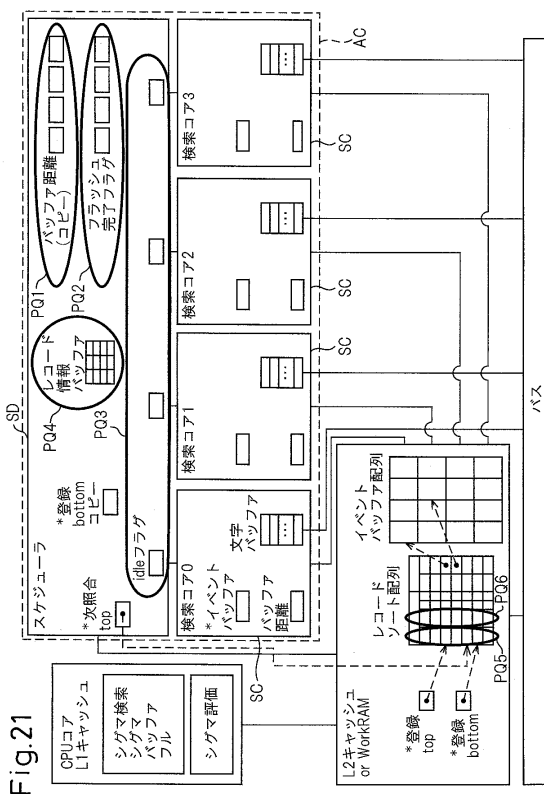
【図19B】



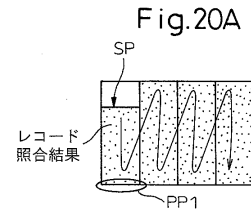
【図19C】



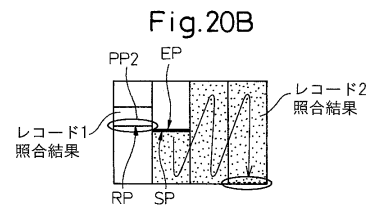
【図21】



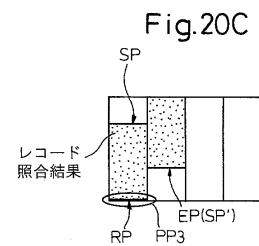
【図20A】



【図20B】

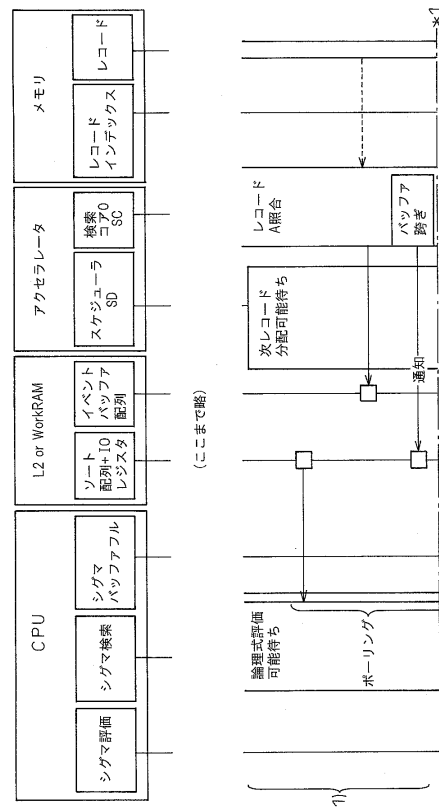


【図20C】

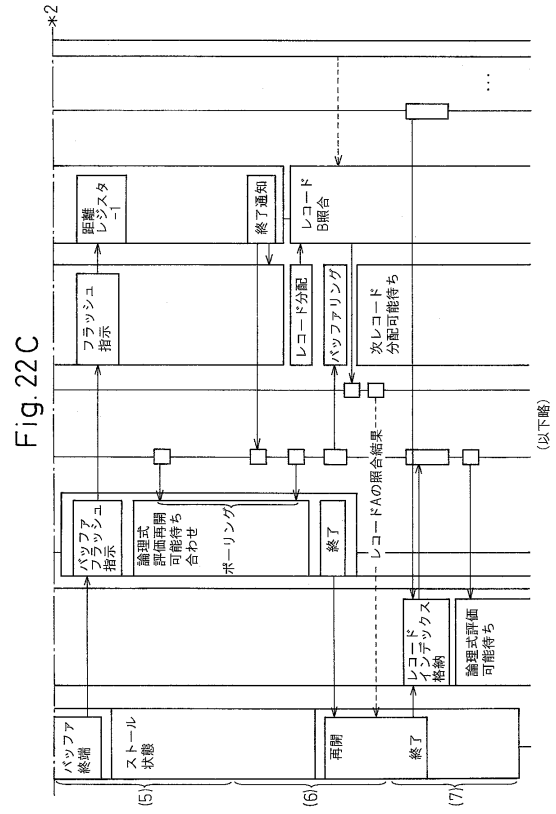


【図22A】

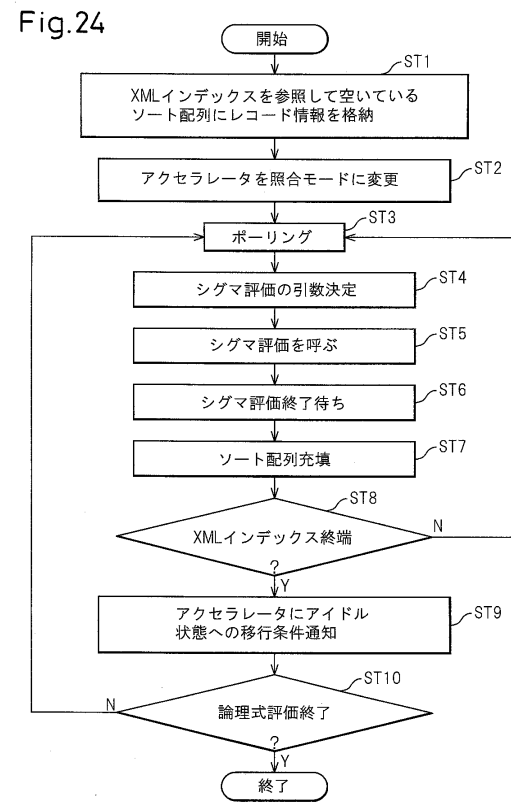
Fig.22A



【図 2 2 C】

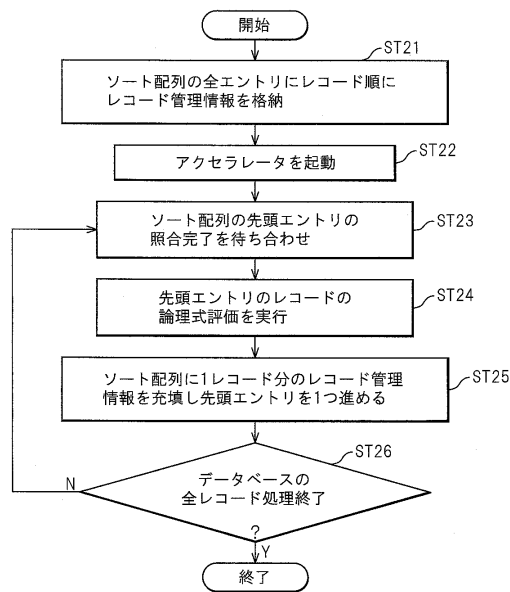


【図 2 4】



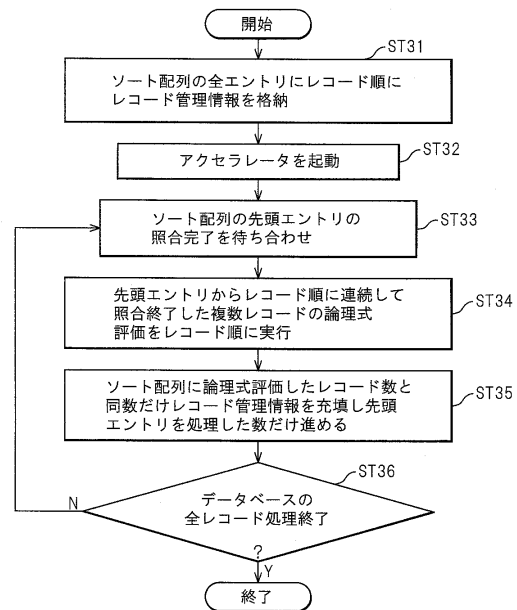
【図 25】

Fig.25



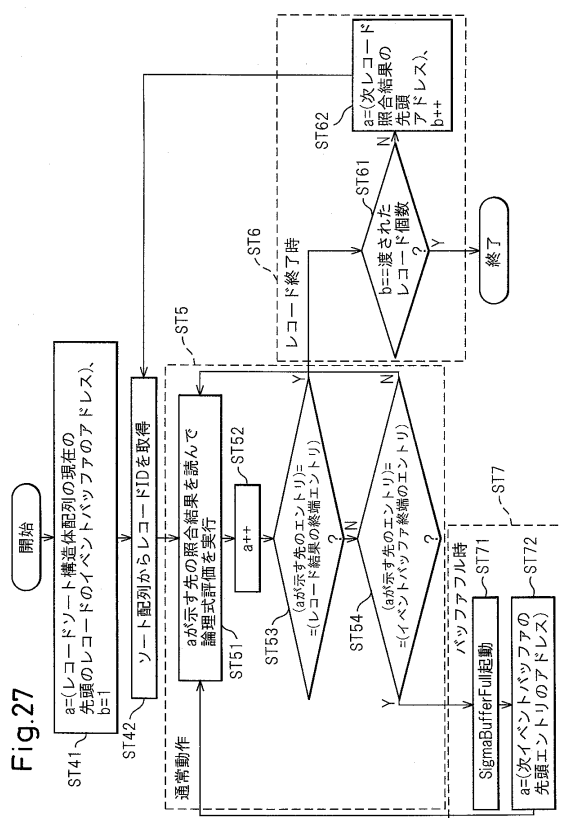
【図 26】

Fig.26



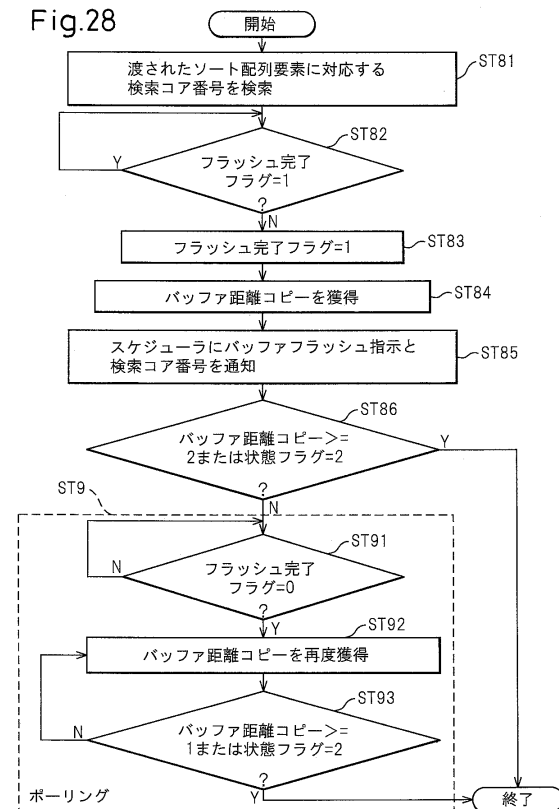
【図 27】

Fig.27



【図 28】

Fig.28



フロントページの続き

(72)発明者 多湖 真一郎

神奈川県川崎市中原区上小田中4丁目1番1号 富士通株式会社内

審査官 野崎 大進

(56)参考文献 特開平6-96124 (JP, A)

国際公開第2004/084095 (WO, A1)

特開2006-228254 (JP, A)

特開2005-242672 (JP, A)

長倉 浩士, データベースエンジンShunsakuによるAsISコンセプトの実現, FUJITSU, 日本, 富士通株式会社, 2006年 4月 1日, VOL. 57、NO. 2, pp.134-139.

(58)調査した分野(Int.Cl., DB名)

G06F 17/30

JSTPlus(JDreamII)