

(19) 日本国特許庁(JP)

(12) 特 許 公 報(B2)

(11) 特許番号

特許第4741134号

(P4741134)

(45) 発行日 平成23年8月3日(2011.8.3)

(24) 登録日 平成23年5月13日(2011.5.13)

(51) Int.Cl.

F I

H04L 12/56 (2006.01)

H04L 12/56 I O O Z

請求項の数 42 (全 50 頁)

(21) 出願番号 特願2001-542932 (P2001-542932)
 (86) (22) 出願日 平成12年12月8日 (2000.12.8)
 (65) 公表番号 特表2003-516660 (P2003-516660A)
 (43) 公表日 平成15年5月13日 (2003.5.13)
 (86) 国際出願番号 PCT/CA2000/001443
 (87) 国際公開番号 W02001/043345
 (87) 国際公開日 平成13年6月14日 (2001.6.14)
 審査請求日 平成19年12月7日 (2007.12.7)
 (31) 優先権主張番号 60/170,232
 (32) 優先日 平成11年12月10日 (1999.12.10)
 (33) 優先権主張国 米国 (US)
 (31) 優先権主張番号 60/212,966
 (32) 優先日 平成12年6月21日 (2000.6.21)
 (33) 優先権主張国 米国 (US)

(73) 特許権者 509196822
 エスエイテック・グループ・ユー・ビー・
 リミテッド・ライアビリティ・カンパニー
 SATECH GROUP A. B. L
 I M I T E D L I A B I L I T Y C O
 M P A N Y
 アメリカ合衆国、19808 デラウェア
 州、ウィルミントン、センタービル・ロー
 ド、2711、スイート・400
 (74) 代理人 100079108
 弁理士 稲葉 良幸
 (74) 代理人 100109346
 弁理士 大貫 敏史

最終頁に続く

(54) 【発明の名称】 最長一致アドレスルックアップのための方法および装置

(57) 【特許請求の範囲】

【請求項 1】

サブツリーインデックスによってアクセスされる複数個のサブツリーエントリと、
 第1のサブツリーエントリに記憶される稠密サブツリー記述子とを含み、前記稠密サブ
 ツリー記述子は、稠密サブツリーの各葉ごとに値の表示を含み、さらに、
 第2のサブツリーエントリに記憶される複数個の疎サブツリー記述子を含み、各疎サブ
 ツリー記述子は少なくとも1つのノード記述子を含む、サブツリーの葉のための値を与
 えるルックアップテーブル。

【請求項 2】

前記ノード記述子は、疎サブツリーの一組の葉のための共通値を識別する、請求項 1 に
 記載のルックアップテーブル。

【請求項 3】

前記稠密サブツリー記述子において「1」にセットされたビットの数に依存して前記稠
 密サブツリーの葉に対応する値を選択するマッピング論理をさらに含む、請求項 1 に記載のル
 ックアップテーブル。

【請求項 4】

サブツリーエントリに複数個の疎サブツリー記述子を記憶するために設けられたビット
 の数は、前記稠密サブツリー記述子におけるビットの数に等しい、請求項 1 に記載のルッ
 クアップテーブル。

【請求項 5】

前記ノード記述子は、ランレングス符号化を用いて符号化される、請求項 1 に記載のルックアップテーブル。

【請求項 6】

前記ノード記述子におけるビットの数は、サブツリー葉におけるビットの数に 1 を加えたものに等しい、請求項 5 に記載のルックアップテーブル。

【請求項 7】

複数のサブツリーの葉のための値は、「1」にセットされた最初の最下位ビットの位置に依存して前記ノード記述子に符号化される、請求項 6 に記載のルックアップテーブル。

【請求項 8】

前記ノード記述子におけるビットの数は 9 であり、前記サブツリー葉におけるビットの数は 8 である、請求項 6 に記載のルックアップテーブル。

10

【請求項 9】

前記サブツリー葉によってサーチ可能な連想メモリと、

選択された疎サブツリー記述子において前記ノード記述子を選択し、前記ノード記述子を 8 ビット符号化値に変換し、前記 8 ビット符号化値を前記連想メモリに記憶する変換論理とをさらに含む、請求項 8 に記載のルックアップテーブル。

【請求項 10】

前記ノード記述子は、「1」にセットされた前記最初の最下位ビットの右側の最下位ビットを「ドントケア」にセットし、「1」にセットされた前記最初の最下位ビットを除去することにより変換される、請求項 9 に記載のルックアップテーブル。

20

【請求項 11】

サブツリーインデックスによってアクセスされる複数のサブツリーエントリと、

第 1 のサブツリーエントリに記憶される稠密サブツリー記述子とを含み、前記稠密サブツリー記述子は、稠密サブツリーの各葉ごとに値の表示を含み、さらに、

第 2 のサブツリーエントリに複数の疎サブツリー記述子を記憶するための手段を含み、各疎サブツリー記述子は少なくとも 1 つのノード記述子を含む、サブツリーの葉のための値を与えるルックアップテーブル。

【請求項 12】

前記ノード記述子は、疎サブツリーの一組の葉のための共通値を識別する、請求項 11 に記載のルックアップテーブル。

30

【請求項 13】

前記稠密サブツリー記述子において「1」にセットされたビットの数に依存して前記稠密サブツリーの葉に対応する値を選択するための手段をさらに含む、請求項 12 に記載のルックアップテーブル。

【請求項 14】

サブツリーエントリに複数の疎サブツリー記述子を記憶するために設けられたビットの数は、前記稠密サブツリー記述子におけるビットの数に等しい、請求項 12 に記載のルックアップテーブル。

【請求項 15】

前記ノード記述子は、ランレングス符号化を用いて符号化される、請求項 12 に記載のルックアップテーブル。

40

【請求項 16】

前記ノード記述子におけるビットの数は、サブツリー葉におけるビットの数に 1 を加えたものに等しい、請求項 15 に記載のルックアップテーブル。

【請求項 17】

複数のサブツリーの葉のための値は、「1」にセットされた最初の最下位ビットの位置に依存して前記ノード記述子に符号化される、請求項 16 に記載のルックアップテーブル。

【請求項 18】

前記ノード記述子におけるビットの数は 9 であり、前記サブツリー葉におけるビットの

50

数は 8 である、請求項 17 に記載のルックアップテーブル。

【請求項 19】

前記サブツリー葉によってサーチ可能な連想メモリと、

選択された疎サブツリー記述子において前記ノード記述子を選択し、前記ノード記述子を 8 ビット符号化値に変換し、前記 8 ビット符号化値を前記連想メモリに記憶する変換するための手段とをさらに含む、請求項 18 に記載のルックアップテーブル。

【請求項 20】

前記ノード記述子は、「1」にセットされた前記最初の最下位ビットの右側の最下位ビットを「ドントケア」にセットし、「1」にセットされた前記最初の最下位ビットを除去することにより変換される、請求項 19 に記載のルックアップテーブル。

10

【請求項 21】

サブツリーインデックスによって複数のサブツリーエントリにアクセスするステップと、

第 1 のサブツリーエントリに稠密サブツリー記述子を記憶するステップとを含み、前記稠密サブツリー記述子は、稠密サブツリーの各葉ごとに値の表示を含み、さらに、

第 2 のサブツリーエントリに複数の疎サブツリーを記憶するステップを含み、各疎サブツリー記述子は少なくとも 1 つのノード記述子を含む、ルックアップテーブルに実現されるサブツリーの葉のための値を与えるための方法。

【請求項 22】

前記ノード記述子は、疎サブツリーの一組の葉のための共通値を識別する、請求項 21 に記載の方法。

20

【請求項 23】

前記稠密サブツリー記述子において「1」にセットされたビットの数に依存して前記稠密サブツリーの葉に対応する値を選択するステップをさらに含む、請求項 21 に記載の方法。

【請求項 24】

サブツリーエントリに複数の疎サブツリー記述子を記憶するために設けられたビットの数は、前記稠密サブツリー記述子におけるビットの数に等しい、請求項 21 に記載の方法。

【請求項 25】

前記ノード記述子は、ランレングス符号化を用いて符号化される、請求項 21 に記載の方法。

30

【請求項 26】

前記ノード記述子におけるビットの数は、サブツリー葉におけるビットの数に 1 を加えたものに等しい、請求項 25 に記載の方法。

【請求項 27】

複数のサブツリー葉のための値は、「1」にセットされた最初の最下位ビットの位置に依存して前記ノード記述子に符号化される、請求項 26 に記載の方法。

【請求項 28】

前記ノード記述子におけるビットの数は 9 であり、前記サブツリー葉におけるビットの数は 8 である、請求項 27 に記載の方法。

40

【請求項 29】

前記サブツリー葉によってサーチ可能な連想メモリを設けるステップと、

選択された疎サブツリー記述子において前記ノード記述子を選択するステップと、

前記ノード記述子を 8 ビット符号化値に変換するステップと、

前記 8 ビット符号化値を前記連想メモリに記憶するステップとをさらに含む、請求項 28 に記載の方法。

【請求項 30】

前記ノード記述子は、「1」にセットされた前記最初の最下位ビットの右側の最下位ビット「ドントケア」にセットし、「1」にセットされた前記最初の最下位ビットを除去す

50

ることにより変換される、請求項 29 に記載の方法。

【請求項 31】

複数レベルルックアップテーブル中のキーに対応するルートを検索するための方法であって、

サブツリーインデックスによってサブツリーエントリを選択するステップを含み、前記サブツリーエントリは複数個の疎サブツリー記述子を含み、さらに

選択されたサブツリーエントリ中の前記複数個の疎サブツリー記述子のうち 1 つを選択するステップを含み、前記疎サブツリー記述子は複数個のノード記述子を含み、さらに
選択された前記疎サブツリー記述子中の前記ノード記述子をマスクされた値に変換するステップと、

前記マスクされた値を最長プレフィックスの順序でメモリにロードするステップと、
 前記キーの部分の最長プレフィックス一致を求めて前記メモリを検索するステップと

、
 前記最長プレフィックス一致のアドレスに依存して別のメモリに記憶される前記ルートを選択するステップとを含む、方法。

【請求項 32】

前記メモリは連想メモリである、請求項 31 に記載の方法。

【請求項 33】

前記ノード記述子は、疎サブツリーの一組の葉のための共通値を識別する、請求項 32 に記載の方法。

【請求項 34】

前記ノード記述子は、ランレングス符号化を用いて符号化される、請求項 33 に記載の方法。

【請求項 35】

前記ノード記述子におけるビットの数は、サブツリー葉におけるビットの数に 1 を加えたものに等しい、請求項 34 に記載の方法。

【請求項 36】

複数のサブツリー葉のためのマスクされた値は、「1」にセットされた最初の最下位ビットの位置に依存して前記ノード記述子に符号化される、請求項 35 に記載の方法。

【請求項 37】

前記ノード記述子におけるビットの数は 9 であり、前記マスクされた値におけるビットの数は 8 である、請求項 36 に記載の方法。

【請求項 38】

前記ノード記述子は、「1」にセットされた前記最初の最下位ビットの右側の最下位ビットを「ドントケア」にセットし、「1」にセットされた前記最初の最下位ビットを除去することにより変換される、請求項 37 に記載の方法。

【請求項 39】

前記ルックアップテーブルは、少なくとも 1 つの稠密サブツリーと、複数個の疎サブツリーとを含み、各疎サブツリー記述子に含まれるノード記述子の数は前記疎サブツリーの葉の数より少ない、請求項 1 に記載のルックアップテーブル。

【請求項 40】

前記ルックアップテーブルは、少なくとも 1 つの稠密サブツリーと、複数個の疎サブツリーとを含み、各疎サブツリー記述子に含まれるノード記述子の数は前記疎サブツリーの葉の数より少ない、請求項 11 に記載のルックアップテーブル。

【請求項 41】

前記ルックアップテーブルは、少なくとも 1 つの稠密サブツリーと、複数個の疎サブツリーとを含み、各疎サブツリー記述子に含まれるノード記述子の数は前記疎サブツリーの葉の数より少ない、請求項 21 に記載の方法。

【請求項 42】

各疎サブツリー記述子に含まれるノード記述子の数は前記疎サブツリーの葉の数より少

10

20

30

40

50

ない、請求項 31 に記載の方法。

【発明の詳細な説明】

【0001】

【発明の背景】

インターネットはルータで接続された 1 セットのネットワークである。ルータは、受信したデータパケットを転送すべき次のホップを、各々の可能なデスティネーションネットワークごとに示すルーティングテーブルを維持する。次のホップとは、別のルータまたは最終的なデスティネーションであろう。

【0002】

ルータのポートで受信されるインターネットプロトコル (IP) データパケットは IP デスティネーションアドレスを含む。IP デスティネーションアドレスは IP データパケットの最終デスティネーションである。現在、IP バージョン 4 (IPv4) および IP バージョン 6 (IPv6) の 2 つの IP バージョンが存在する。IPv4 は、IP デスティネーションアドレスを記憶する、データパケットに含まれる IP ヘッダに 32 ビットのフィールドを設ける。ルータは、IP ヘッダに記憶された IP デスティネーションアドレスに依存して、接続された受信データパケットを次のループのルータかまたは、デスティネーションがローカルネットワークであれば最終的なデスティネーションに転送する。

【0003】

32 ビット IPv4 デスティネーションアドレスは 40 億の可能なルートをもたらす。インターネットルータは典型的に、40 億の可能なルートのうち 5 万を記憶する。しかしながら、記憶されるルート数は、インターネットの発展および IPv6 の広がりによって増加している。

【0004】

もともと、IP アドレス空間は 3 つのクラスの IP アドレス、すなわち A、B および C に分割されていた。各々の IP アドレス空間はネットワークアドレスおよびホストアドレスに分割された。クラス A は、126 のネットワークおよびネットワーク当り 1600 万個のホストを許容した。クラス B は、ネットワーク当り 64,000 個のホストを備えて、16382 個のネットワークを許容し、クラス C は、ネットワーク当り 256 個のホストを備えて、200 万個のネットワークを許容した。しかしながら、IP アドレス空間を異なるクラスに分割することにより、利用可能な IP アドレスの数が減少してしまった。クラス C はネットワーク当り最大で 256 個のホストしか許容しないが、これは大抵の組織にとっては少なすぎる。したがって、大抵の組織にクラス B アドレスを割当てて、64,000 個のホストアドレスをとった。このホストアドレスは、それらを割当てられた組織が使用しなくても、他の組織が使用不可能なものである。クラス B の IP アドレスを有する組織のホストのすべては、たとえば、27.32.x.x などの、16 の最上位ビット (MSB) に同じネットワークアドレスを記憶する。

【0005】

クラスレス・インタードメイン・ルーティング (CIDR) は、未使用の IP ホストアドレスを自由化するために導入された。残余の未使用のネットワークは、サイズ可変のブロックで組織に割当てられる。500 個のアドレスを必要とする組織は 500 個の連続するアドレスを入手する。たとえば、ある組織に、128.32.x.x で始まる 500 個の利用可能なアドレスを割当て可能である。ルータが記憶するルート数は、クラスレス・インタードメイン・ルーティングの導入以来増加した。クラスレス・インタードメイン・ルーティングは、一致するネットワークアドレスをサーチして IP デスティネーションアドレスの対応する次のホップを見出す代わりに、対応するルートを見出すのに最長プレフィックス一致を必要とする。たとえば、サーチは、たとえば 128.x.x.x などのクラス B の IP アドレスの 16 MSB の後はもはやストップできない。なぜなら、128.32.4.x.x は別の組織に割当てられ得るからであり、これには次の異なるホップを必要とする。

【0006】

キーに対して最長プレフィックス一致をサーチするための1つの方法は、バイナリツリーサーチの使用によるものである。バイナリツリーサーチは、1ビットずつ32のレベルまで32ビット入力を一致させ、32ビットのキーに一致するエントリを見出すには32のサーチが必要である。一致をサーチするための別の方法はパトリシアツリーの使用によるものである。パトリシアツリーは、バイナリツリーの葉までにエントリが存在しなければ、必要なサーチ数を減らす。

【0007】

IPデスティネーションアドレスと関連する次のホップを効率的にサーチするためのまた別の方法が、1998年5月11日に出願された、ブロードニクら(Brodnick et al.)による「高速ルーティングルックアップのための方法およびシステム」(“Method and System for Fast Routing Lookups”)と題された、PCT出願連続番号PCT/SE98/00854に記載されている。ブロードニクが記載する方法は、重複するルートを記憶しないことで、記憶される次のホップの数を減じる。次のホップの数を減じることにより、メモリの必要量を低減し、それによりルートルックアップテーブルを高速キャッシュメモリに記憶できるようになる。

【0008】

ブロードニクらは、バイナリツリーを3レベルに分割する。バイナリツリーを3つのレベルに分割することにより、サーチ数を3に減らした。第1のレベルのインデックス付けされたエントリは、ルートをエントリからとる場合、サーチを第1のレベルで終了可能であるかまたは、IPデスティネーションアドレスのさらなる部分を用いて、その後のレベルまでサーチを継続しなければならないかを示す。

【0009】

図1Aは、バイナリツリーの第1のレベルを表わす先行技術の64K(65536)ビットマップを図示する。64Kビットマップ30は、深度16のバイナリツリーの葉またはノード44を表わし、ノード44当り1ビットである。ビットマップは長さ16のビットマスクに分割される。64Kビットマップには $2^{12} = 4096$ のビットマスクが存在する。図1Aに1つのビットマスクが示される。ビットマップ30中のビットは、ノード44に対応するポインタの配列に記憶されたルートインデックスまたはサブツリーが存在すれば、‘1’にセットされる。ノードが前のノード44とルートエントリを共有していれば、ビットマップ30中のビットは、‘0’にセットされる。

【0010】

図1Bは、キャッシュメモリに実現される先行技術のルックアップテーブルを図示する。ルックアップテーブルは、コードワード36の配列、ベースインデックス34の配列およびマップテーブル40を含む。図1Bには32ビットのIPアドレス38も示される。コードワード46は、ビットマップ30(図1A)の各ビットマスクごとにコードワード36の配列に記憶される。コードワード46は6ビットの値46aおよび10ビットのオフセット46bを含む。ベースインデックス42は、コードワード36の配列中の4つのコードワード46ごとに、ベースインデックス34の配列に記憶される。

【0011】

コードワード36の配列、ベースインデックス34の配列およびマップテーブル40を用いて、ポインタ(図示せず)の配列中のポインタを選択する。ポインタは、ルートインデックスまたはインデックスを記憶してさらなるサーチを行なう。

【0012】

ポインタの配列中のポインタ群は、コードワード36の配列中のコードワード46およびベースインデックス34の配列中のベースインデックス42を選択することによって選択される。コードワード46は、IPアドレス38の最初の12ビット50を用いて選択される。ベースインデックス42は、IPアドレス38の最初の10ビット48を用いて選択される。マップテーブル32を用いて、ポインタ群の中の正しいポインタが選択される。

【0013】

10

20

30

40

50

選択されたコードワード 36 中の 10 ビットの値 46 b は、マップテーブル 32 へのインデックスである。マップテーブル 32 は、ビットマスク内のビット数を 4 ビットのオフセットにマッピングする。オフセットは、ポインタのアレイの中の選択されたポインタ群内のポインタを特定する。10 ビットの値 46 b はマップテーブル 32 中の列を選択し、IP アドレス 52 のビット 19 : 16 は 4 ビットのオフセット 54 を選択する。

【0014】

このように、ポインタのサーチには、以下のキャッシュメモリアクセスが必要である。すなわち、(1) 16 ビットのコードワード 46 を読出す；(2) 16 ビットのベースアドレス 42 を読出す；(3) マップテーブル 32 から 4 ビットのオフセット 54 を読出す；(4) ポインタインデックスがベースアドレス 42、コードワードオフセット 46 a および 4 ビットのオフセット 54 の和であるポインタインデックスでポインタを読出す。

10

【0015】

バイナリツリーの各レベルで同じメモリアクセスが必要である。したがって、3 レベルのサーチには 12 のメモリアクセスが必要である。

【0016】

IP 宛先アドレスと関連付けられる次のホップのために効率よくサーチを行なうための方法が、1998 年 5 月 11 日に出版されたプロドニックらによる「高速ルーティングルックアップのための方法およびシステム」と題する PCT 出願連続番号第 PCT/SE98/00854 号に記載される。プロドニックらは、疎であるサブツリー；すなわち、8 個未満のルートが記憶されるサブツリーに必要とされるバイトの数を低減するための方法を記載する。24 バイトのメモリが、疎サブツリーのための 8 個のルートを記憶するために割当てられる。24 バイトは、サーチキーに対応する値を記憶する 8 個の 8 ビットルートエントリのアレイと、各ルートエントリごとのそれぞれの 16 ビットのポインタとを含む。

20

【0017】

値は、降順にメモリのアレイに記憶される。アレイは、サーチキーより小さいまたは等しい値を見つけるためにサーチされる。メモリへの複数アクセスが必要とされる。多くとも 7 個のメモリアクセスが、サーチキーに対応する値を見つけるために必要とされ得る。

【0018】

【発明の概要】

30

複数の疎サブツリー記述子がメモリのサブツリーエントリに記憶されることを可能にするルックアップテーブルが提供される。ルックアップテーブルは、サブツリーインデックスによってアクセスされる複数個のサブツリーエントリを含む。稠密サブツリー記述子が、第 1 のサブツリーエントリに記憶される。稠密サブツリー記述子は、稠密サブツリーの各葉ごとに値の表示を含む。複数個の疎サブツリー記述子が、第 2 のサブツリーエントリに記憶される。疎サブツリー記述子の各々は、少なくとも 1 つのノード記述子を含む。

【0019】

ノード記述子は、疎サブツリーの一組の葉のための共通値を識別する。メモリは、疎サブツリー記述子および稠密サブツリー記述子の両方を記憶する。稠密サブツリー記述子が第 1 のサブツリーエントリに記憶される。稠密サブツリー記述子は、稠密サブツリーの葉当り 1 ビットを含む。稠密サブツリー記述子は、稠密サブツリーの葉のための値を示す。複数個の疎サブツリー記述子が第 2 のサブツリーエントリに記憶される。各疎サブツリー記述子は、少なくとも 1 つのノード記述子を含む。ノード記述子は、疎サブツリーの一組の葉のための共通値を識別する。

40

【0020】

マップ論理が、稠密サブツリー記述子において「1」にセットされたビットの数に依存して稠密サブツリーの葉に対応する値を選択する。サブツリーエントリに疎サブツリー記述子を記憶するために設けられたビットの数は、稠密サブツリー記述子におけるビットの数に等しい。

【0021】

50

ノード記述子は、ランレングス符号化を用いて符号化される。ノード記述子におけるビットの数は、サブツリー葉におけるビットの数に1を加えたものに等しい。複数のサブツリー葉のための値は、「1」にセットされた最初の最下位ビットの位置に依存してノード記述子に符号化される。ノード記述子におけるビットの数は9であり、サブツリー葉におけるビットの数は8である。

【0022】

サブツリー葉によってサーチ可能な連想メモリが設けられる。変換論理が、選択された疎サブツリー記述子においてノード記述子を選択し、ノード記述子を8ビット符号化値に変換し、8ビット符号化値を連想メモリに記憶する。ノード記述子は、「1」にセットされた最初の最下位ビットの右側の最下位ビットを「ドントケア」にセットし、「1」にセットされた最初の最下位ビットを除去することにより変換される。

10

【0023】

この発明の以上およびその他の目的、特徴および利点は、添付の図面に図示されるように、この発明の好ましい実施例の以下のより特定の説明から明らかになるであろう。図面中の同じ参照番号は異なる図面を通じて同じ部分を示す。図面は、この発明の原則を図示するのに供される代わりに、縮尺を変更したり、強調したりということを必ずしも行わなくてよい。

【0024】

【詳細な説明】

この発明の好ましい実施例の説明は以下のとおりである。

20

【0025】

デンスモード

図2Aは、この発明の原則に従う最長一致プレフィックスルックアップテーブル100を図示する。ルックアップテーブル100はキー104のルートインデックス102を与える。ルートインデックス102を用いて、IPデスティネーションアドレスに向けた次のホップにアクセスする。図2Aに示された実施例では、キー104は32ビット幅であるが、キー104は32ビットに限定されるものではない。ルックアップテーブル100は3つのマップ106a-cを含む。各々のマップ106a-cは別個にアドレス指定されたメモリを含む。キー104に対応するデフォルトルートインデックスまたはルートインデックス102は、マップ106a-cのうち1つの中の場所に記憶される。各マップからのマップ出力110a-cは、多数のマップをサーチする必要がある場合、キーについてすべてのマップ106a-cにアクセスするまで、遅延メモリ150a-cに記憶される。

30

【0026】

マルチプレクサ108は、マルチプレクサ108の入力に転送されるマップ出力110a-cの1つをルートインデックス102として選択する。マップ出力110a-cは、マップ出力110a-cの最上位ビット(MSB)に依存して選択される。マップ出力110a-cがルートインデックス102を含む場合のみ、マップ出力110a-cのMSBが「1」となる。

【0027】

40

図2Bは、図2Aに示されたルックアップテーブル100のマップ106a-cに記憶されたエントリのバイナリツリー表示を図示する。図2Bは図2Aと関連して説明される。32ビットのキー104は32レベルのバイナリツリーとして表わすことができる。バイナリツリー実現例は、1ビットずつ32レベルまでサーチするには、32のサーチを必要とする。サーチ数を減らすため、バイナリツリーの32のレベルは3つのマップレベル112a-cに分割され、各マップレベル112a-cはマップ106a-c(図2A)に対応する。マップレベル__1 112aは、バイナリツリーの32のレベルのうち最初の16個を含む。しかしながら、簡潔さのため、図2Bには16のレベルのうち5つしか示されない。マップレベル__2 112bは、32レベルのバイナリツリーの次の8レベルを含み、図2Bには8つのレベルのうち3つが示される。マップレベル__3は、32レベ

50

ルのバイナリツリーのうち最後の8レベルを含み、図1Bには8つのレベルのうち3つが示される。各マップレベル112a-cは複数のノードを含む。16レベル(キー104の16のMSB)がマップレベル-1 112aにあり、8レベルがマップレベル__2 112bにありかつ8レベルがマップレベル__3にあるように32レベルを分割することが、現在のメモリ技術においては最適であるように思われるが、この発明はこの構成に限定されるものではない。

【0028】

キー104の最初の16ビットに対して1ビットずつ別個の16のサーチを行なう代わりに、キー104の最初の16ビットと関連するルートインデックス102をマップ106a(図2A)に記憶する。マップ106a(図2A)は、キー104の最初の16ビットのMSBによって直接にインデックス付けされる。次のマップ106bは、前のマップ106aが、キー104と関連する次のホップ情報にアクセスするのに用いるルートインデックス102を記憶しているか否かに依存して、サーチされる。

【0029】

図2Bに示されるように、マップレベル__1 112aに示されるノードまたは葉は、それぞれr0およびr1と標識付けられた2つのルート114、116と、それぞれs0およびs1と標識付けられたマップレベル__2 112bへの2つのポインタ130⁴および130²³を含む。各ルート114、116ごとのルートインデックス102はL1マップ106aに記憶される。また、L2マップ106bに向けたアドレスポインタ120はサブツリーインデックス130⁴に対して記憶され、L2マップ106bに向けたアドレスポインタ(図示せず)はサブツリー130²³に対して記憶される。サブツリーインデックス130⁴に対して、マップ106a中のマップエントリ140⁴に記憶されるアドレスポインタ120は、キー104と関連のルートインデックス102を見出すためには、次のレベルのサーチが必要であることを示す。

【0030】

ツリー中のいずれのノードの値も、根114からの経路をトレースすることによって判定することができる。バイナリツリー中の各ノードは、右側の子および左側の子の2つの子を有して示される。親ノードが'1'ならば右側の子が選択される。親ノードが'0'ならば左側の子が選択される。根114からノード116への経路をトレースすると、r1は、'010'にセットされたMSBを有するすべてのキーに対して、L1マップ106a中にルートインデックス102として記憶される。根ノード114からs0ノード130⁴への経路をトレースすると、s0は、'00011'にセットされたMSBを有するすべてのキーに対してL1マップ106aに記憶される。

【0031】

L1マップ106aはダイレクト・マップト・マップであり、マップレベル__1 112aのボトムレベルの各ボトムレベルノードまたは葉ごとにルートインデックス102を記憶する。マップレベル__1 112aのボトムレベルは、32レベルバイナリツリーの16番目のレベルである。16番目のレベルは64Kのノードを有する。しかしながら、例示の目的のため、マップレベル__1 112aのボトムレベルは、32レベルのバイナリツリーのレベル-5として示される。L1マップ106aに示されるルートインデックス102は、マップレベル__1 112aのレベル-5の130¹-130³²ノードに対応する。根ノード114からレベル__5のノード130¹、130²、130³への経路をトレースすると、ルートインデックス102はr0である。したがって、r0は、L1マップ106aの場所140¹、140²、140³に、すなわち、インデックス00000、00001および00010に記憶される。ノード130⁴はサブツリーインデックスs0を記憶し、したがって、s0は、アドレス00011に、L1マップ106aの中の場合140⁴に記憶される。同様に、レベル-5のノード130⁵-130⁸に対するルートインデックス102はr0であり、したがって、L1マップ106aの中のアドレス00100、00101、00110および00111の場所140⁵、140⁶、140⁷、140⁸はr0を記憶する。レベル-5のノード130⁹-130¹²に対するルートインデ

10

20

30

40

50

ックス102はr1であり、したがって、L1マップ106aの中のアドレス01000および010001の場所140⁹、140¹⁰はr1を記憶する。

【0032】

L1マップ106a中の各々の場所は、直接にもしくはレベル-5のノード300¹⁻³²の親を介してレベル-5のノード300¹-300³²に割当てられるルートインデックス102または次のマップ106b-cへのアドレスポインタを記憶する。マップレベル__3106cは、32レベルのバイナリツリーのボトムレベルで、ノード138のh0およびノード140のh1の2つのホストノードを含む。ホストノードのサーチは、キー104のすべてのビットのサーチを必要とする。図2Aに示されるように、h0に対するルートインデックス102は、L3__マップ106c中の場所146⁴に記憶される。L1マップ106aと異なり、L2マップ106bおよびL3マップ106cは直接にマッピングされない。

【0033】

マップ106bおよび106cでは、ルートインデックス102は各々の可能な入力ごとに記憶されない。ノードに対するルートインデックス102がマップ106b-cに以前に記憶されたルートインデックス102と異なる場合にのみ、ルートインデックス102が記憶される。マップレベル__2112bに示される第1のサブツリーAのレベル-3ノードを見ると、ノード132¹およびノード132²に対するルートインデックス102はr0であり、したがって、r0に対するルートインデックスは、L2マップ106b中のノード132¹およびノード132²の両者について、場所142¹に記憶される。ノード132²に対するサブツリーインデックスs0は場所142²に記憶される。レベル-3のノード132⁴およびレベル-3のノード132⁵および132⁶と関連のルートインデックス102はr0であり、これは、前のノード132²に対して記憶されたs0とは異なる。したがって、r0はL2マップ106b中の次の場所142³に記憶される。ルートr2は、ノード132⁷に対して、L2マップ106b中の場所142⁴に記憶される。なぜなら、ノード132⁷は前のノード132⁶と同じルートを共有しないからである。サブツリーインデックスs3は次のレベル-3のノード132⁷に対して記憶される。したがって、s3はL2マップ106b中の場所142⁵に記憶される。前のノードからルート変更があるときにだけルートインデックスを記憶することにより、ルートインデックス102を記憶するのに必要なメモリが低減される。示されるように、マップレベル__2112bの第1のサブツリーAのレベル-3の8つのノード132¹⁻⁸についてルートインデックスを記憶するためには、L2マップ106b中の場所は5つしか必要でない。ノンダイレクトマップ106b、106cは、図5と関連してより詳細に後述される。

【0034】

図3は、この発明の原則に従う40ビットのキー210のための最長一致プレフィックスルックアップテーブル200を図示する。1つの実施例では、40ビットのキーは、8ビットのプレフィックスおよび32ビットのIPアドレスを含む。8ビットのプレフィックスは、32ビットのIPアドレスと関連する仮想私設網(VPN)識別子であり得る。ルックアップテーブル200は4つのマップ106a-dを含む。マップ106aは、図2Aと関連して説明されたように、ダイレクト・マップト・マップである。マップ106b-dはインダイレクトマップである。マップ106aは、40ビットのキー210の16のMSBに対応する、L2マップ106bに対するサブツリーインデックスまたはルートインデックス102を記憶する。したがって、L1マップは、第1のマップレベル112a(図2B)中の64Kのノードの各々ごとに1つの、64Kの可能な場所を有する。L1マップ106a中の対応する場所に記憶されるL1マップエントリデータ220aは、パイプライン208およびL2インダイレクトマップ106bに転送される。L1マップエントリデータ220aが、キー210bの次の8ビットを用いる次のレベルのサーチが必要であることを示せば、キー210bの次の8ビットおよびL1マップエントリデータ220aに依存して、L2インダイレクトマップ106bでサーチが行なわれる。

【 0 0 3 5 】

第2のレベルのサーチの結果がL 2 マップエントリデータ 2 2 0 b 上に与えられ、これはパイプライン 2 0 8 および L 3 インダイレクトマップ 1 0 6 c に転送される。第3のレベルのサーチは、キー 2 1 0 c の次の8ビットおよび L 2 マップエントリデータ 2 2 0 b に依存して、L 3 インダイレクトマップ 1 0 6 c で行なわれる。

【 0 0 3 6 】

サーチの結果は、L 3 マップエントリデータ 2 2 0 c 上を、パイプライン 2 0 8 および L 4 インダイレクトマップ 1 0 6 d に与えられる。L 3 マップエントリデータ 2 2 0 c は、キー 2 1 0 d の最後の8ビットおよび L 3 マップエントリデータ 2 2 0 c に依存して、L 4 インダイレクトマップ 1 0 6 d で行なわれる。

10

【 0 0 3 7 】

第4のサーチの結果はL 4 マップエントリデータ 2 2 0 d 上に与えられる。キー 2 1 0 に対する最長一致プレフィックスと関連するルートインデックス 1 0 2 は、マップ 1 0 6 a - d のうちの1つの中のただ1つの場所に記憶される。したがって、パイプライン 2 0 8 に転送されるルートインデックス 1 0 2 は、ただ1つのマップエントリデータ 2 2 0 a - d に含まれる。マップ 1 0 6 a - d のうち1つ、たとえばマップ 1 0 6 b にルートインデックス 1 0 2 を見出すと、残余のマップ 1 0 6 c - d のサーチは必要ではなく、マップ 1 0 6 c - d はアクセスされない。パイプライン 2 0 8 は、マップエントリデータ 2 2 0 a - d のうち1つに含まれるルートインデックス 1 0 2 を選択するためのマルチプレクサ 1 0 8 (図 2 A) を含む。たとえば、マップエントリデータ 2 2 0 a - d の M S B は、ルートインデックスが含まれているか否かの表示を与えることができる。

20

【 0 0 3 8 】

マップ 1 0 6 a - d と関連してパイプライン 2 0 8 を用いることにより、異なるキー 2 1 0 による最長一致プレフィックステーブル 2 0 0 の複数サーチを並行して行なうことができる。パイプライン 2 0 8 は、4 0 ビットのルックアップテーブル 2 0 0 の複数のサーチを、必要に応じて、他のマップ 1 0 6 a - d の各々のサーチが完了するまで、4 0 ビットのキー 2 1 0 と関連の各マップ 1 0 6 a - d ごとにマップエントリデータ 2 2 0 a - d を記憶することによって並行して行なって、4 0 ビットキー 2 1 0 に対応するルートインデックスを見出すのを可能にする。したがって、ダイレクト・マップト・マップ 1 0 6 a への単一のメモリアクセスを行なうことにより、受信した I P アドレスに対応するルートインデックスに対するサーチリクエストがルックアップテーブル 2 0 0 に発せられる。別のキーに対応するルートインデックスに対するその後のサーチは、ダイレクト・マップト・マップ 1 0 6 a のための次のメモリアクセスサイクルの中でルックアップテーブル 2 0 0 に発せられ得る。

30

【 0 0 3 9 】

図 4 は、図 3 に示されたダイレクト・マップト・マップ 1 0 6 a に記憶可能なマップエントリのタイプを図示する。図 2 B に示されたバイナリツリー中のいずれのノードに対するマップエントリも、ノーエントリ 3 0 0、ルートエントリ 3 0 2 またはサブツリーエントリ記述子 3 0 4 を記憶することができる。マップエントリ 3 0 0、3 0 2、3 0 4 の各タイプは、サブツリーフラグ 3 0 6 を含む。サブツリーフラグ 3 0 6 の状態は、マップエントリがサブツリーエントリ記述子 3 0 4 であるか否かを示す。サブツリーフラグ 3 0 6 が ' 1 ' にセットされれば、マップエントリはサブツリーエントリ記述子 3 0 4 であり、サブツリーインデックス 3 1 2 を含む。サブツリーインデックス 3 1 2 は、次のノンダイレクト・マップト・マップ 1 0 6 b - d に記憶されるサブツリーエントリ記述子 3 0 4 のアドレスである。サブツリーエントリは図 4 と関連して後述される。サブツリーフラグ 3 0 6 が ' 0 ' ならば、ノーエントリフラグ 3 1 4 をチェックして、マップエントリがノーエントリ 3 0 0 であるかまたはルートエントリ 3 0 2 であるかを判定する。ノーエントリフラグ 3 1 4 が ' 0 ' ならば、エントリはノーエントリ 3 0 0 である。ノーエントリフラグ 3 1 4 が ' 1 ' ならばエントリはルートエントリ 3 0 2 であり、ルートインデックスフィールド 3 1 0 中に、キー 1 0 4 と関連のルートインデックス 1 0 2 (図 3) を記憶する。

40

50

マルチプレクサ 108 (図 2 A) は、サブツリーフラグ 306 を用いて、ルートインデックス 102 (図 3) を含むマップエントリデータ 220 a - d を選択する。

【0040】

図 5 は、図 2 B に示されたマップレベル 2 112 b 中のノードに対応するマップ 106 b を図示する。マップ 106 b は、サブツリーメモリ 400、マップアドレス論理 402 およびサブツリーマップ 418 を含む。マップ 106 a に記憶されるキー 210 a の最初の部分によって選択されたサブツリーインデックス 312 は、サブツリーメモリ 400 に転送される。サブツリーメモリ 400 は、サブツリーインデックス 312 が選択するサブツリーエントリ 404 を含む。サブツリーエントリ 404 はデータフィールド 406 およびポインタフィールド 408 を含む。

10

【0041】

図 2 B に戻って、サブツリーエントリ 404 は、マップレベル 2 112 b に示されるサブツリーのうちの 1 つのボトムレベルに対応する。マップレベル 2 112 b が 8 つのレベルを有すれば、各サブツリー (図示せず) のボトムレベルは、各ノードに 1 つの、最大 256 のルート有する。

【0042】

続いて図 5 で、サブツリーエントリ 404 は、サブツリーのボトムレベル上の各ノードに対応する 256 の可能なルートインデックス 102 (図 3) へのアクセスを提供する。ルートインデックス 102 (図 3) はサブツリーマップ 418 に記憶される。256 の可能なルートインデックスへのアクセスを提供するため、稠密サブツリー記述子がデータフィールド 406 に記憶される。データフィールド 406 は 256 ビット幅であり、サブツリーのボトムレベルで各ノードごとに 1 ビットを与える。データフィールド 406 は、図 6 A および図 6 B と関連してより詳細に後述される。ポインタフィールド 408 は 256 ビット幅であり、16 の 16 ビットポインタの記憶を可能にする。各ポインタは、サブツリーマップ 418 中に、連続した 16 のマップエントリに対するベースアドレスを記憶して、256 のルートインデックスへのアクセスを与える。したがって、ポインタフィールド 408 は、サブツリーのボトムレベルの各ノードごとにサブツリーマップ 418 中のマップエントリへのポインタを間接的に与えることができる。ポインタフィールド 408 は図 6 と関連してより詳細に説明される。

20

【0043】

データフィールド 406 中の稠密サブツリー記述子に記憶されたサブツリーデータ 412 とポインタフィールド 408 に記憶されたサブツリーポインタ 414 とはマップアドレス論理 402 に転送される。マップアドレス論理 402 は、キー 210 b の次の部分 (次の 8 ビット) も受ける。

30

【0044】

マップアドレス論理 402 は、キー 212 b の次の 8 ビットに依存するサブツリーのボトムレベルのノードと関連のマップエントリ、すなわちサブツリーデータ 412 およびサブツリーと関連のサブツリーポインタ 414 のマップアドレス 416 を判定する。マップアドレス 416 は、サブツリーマップ 418 中のマップエントリを選択する。サブツリーマップ 418 は、ダイレクト・マップト・マップ 106 a に対して、図 4 と関連して説明されたのと同じタイプのマップエントリを含む。マップデータエントリ 220 b の中身が、その後のサーチが必要であるか否かを決定する。マップエントリデータ 220 b が、次のマップレベル 112 c (図 2 B) 中に別のサブツリーエントリ 404 が存在することを示すサブツリーインデックス 312 (図 4) を含めば、その後のサーチが必要である。

40

【0045】

キー 210 b の第 2 の部分は、選択されたサブツリーのボトムレベル中のノードを選択する。サブツリーポインタ 414 は、サブツリー中のノードと関連のベースアドレスを選択し、サブツリーデータ 412 は、ベースアドレスと関連のマップエントリのブロック内のオフセットを選択する。マップアドレス論理 402 は、図 7 と関連して後述される。

【0046】

50

図 6 A はサブツリーのバイナリツリー表示である。示されたサブツリーは 5 つのレベルを含む。サブツリーは、3 つのルートインデックス r_1 、 r_2 および r_3 ならびに 2 つのサブツリーインデックス s_0 および s_1 を含む。サブツリーのボトムレベルには 32 個のノード $500^1 - 500^{32}$ が存在する。ボトムレベルの各ノード $500^1 - 500^{32}$ と関連のルートインデックスまたはサブツリーインデックスが以下の表 1 に示される。

【 0047 】

【表 1】

サブツリービット	ルート/サブツリー
00000	r_0
00001	r_0
00010	r_0
00011	r_0
00100	r_1
00101	r_1
00110	r_0
00111	r_0
01000	r_2
01001	s_0
01010	r_2
01011	r_2
01100	r_2
01110	r_2
01111	r_2
1xxxx	r_3

10

20

表 1

【 0048 】

図 6 B は、図 6 A に示されたサブツリーのボトムレベル中のノードに対応する、図 5 に示されたデータフィールド 406 に記憶された稠密サブツリー記述子を図示する。データフィールド 406 は、図 6 A に示されたサブツリーのボトムレベル中の各ノード 500 ごとに 1 ビットの、32 ビットを含む。データフィールド 406 中のビット $502^1 - 502^{32}$ は以下のように割当てられる。前のノードのルートインデックスを用いるならば、データフィールド 406 中のビットを '0' にセットし、サブツリーマップ 418 (図 5) に記憶された次のルートインデックスを用いるならば、'1' にセットして次のマップエントリアドレスを増分する。ルートが特定されなければ、データフィールド 402 中の最初のビットは、マップエントリ 504^1 に記憶されたデフォルトルート r_0 を選択する。特定されたルートがないため、こうして、ビット 502^1 を '0' にセットしてデフォルトルートを選択する。マップエントリ 504^1 に記憶されたデフォルトルート r_0 は、次の 3 つのノード $500^2 - 500^4$ に対して選択され、すなわち、対応するビット $502^2 - 500^4$ はデータフィールド 406 中で '0' にセットされて、 502^1 が用いた前のルートインデックスを用いる。ノード 500^5 ではルート変更がある。

30

40

【 0049 】

マップエントリ 504^2 に記憶される、ノード 500^5 に用いられるルート r_1 はノード 506^6 と共有される。したがって、ビット 502^5 は、ルート変更を示す '1' であり、サブツリーマップ 418 (図 5) 中のマップエントリ 504^2 を選択する。ビット 502^6 は、 502^5 に記憶されるルートインデックスをこのノードに用いるべきであることを示す '0' である。ノード 500^7 に対してはいずれのルートも与えられないため、ルート変更があり、デフォルトルート r_0 を記憶するサブツリーマップ 418 (図 5) 中のマップエントリ 504^3 を必要として、ビット 502^7 に '1' が記憶される。

【 0050 】

50

ノード500⁸は前のノード500⁷と同じルートを共有し、サブツリーマップ418(図5)で新たなマップエントリは必要ない。ノード500⁸に対応するビット502⁸は‘0’にセットされる。ノード500⁹は前のノード500⁸とは異なるルートを有し、サブツリーマップ418(図5)で新たなマップエントリが必要である。ノード500⁹に対応するビット502⁹は‘1’にセットされ、次の連続するメモリ場所中のサブツリーマップ418(図5)に、r2を記憶するマップエントリ504⁴が加えられる。

【0051】

ノード500¹⁰は、前のノード500⁹とは異なるルートを有し、サブツリーマップ418(図5)で新たなルートエントリが必要である。ノード500¹⁰に対応するビット502¹⁰は‘1’にセットされ、s0を記憶するマップエントリ504⁵が、次の連続するメモリ場所中のサブツリーマップ418(図5)に加えられる。

10

【0052】

ノード500¹¹は、前のノード500¹⁰とは異なるルートを有し、サブツリーマップ418(図5)で新たなマップエントリが必要である。ノード500¹¹に対応するビット502¹¹は‘1’にセットされ、r2を記憶するマップエントリ504⁶が、次の連続するメモリ場所中のサブツリーマップ418(図5)に加えられる。

【0053】

ノード500¹²および500¹³は、前のノード500¹¹と同じルートを共有し、サブツリーマップ418(図5)で新たなマップエントリは必要ない。ノード500¹²に対応するビット502¹²およびノード500¹³に対応するビット502¹³はデータフィールド406中で‘0’にセットされる。

20

【0054】

ノード500¹⁴は前のノード500¹³とは異なるルートを有し、サブツリーマップ418(図5)で新たなマップエントリが必要である。ノード500¹⁴に対応するデータフィールド406中のビット502¹⁴は‘1’にセットされ、s1を記憶するマップエントリ504⁷がサブツリーマップ418(図5)に加えられる。ノード500¹⁵は前のノード500¹⁴とは異なるルートを有し、サブツリーマップ418(図5)で新たなマップエントリが必要である。ノード500¹⁵に対応する、データフィールド中のビット502¹⁵は‘1’にセットされ、サブツリーマップ418(図5)に、r2を記憶するマップエントリ504⁸が加えられる。ノード500¹⁶は前のノード500¹⁵と同じルートを共有し、サブツリーマップ418(図5)で新たなマップエントリは必要ない。

30

【0055】

ノード500¹⁷は前のノード500¹⁶とは異なるルートを有し、サブツリーマップ418(図5)で新たなマップエントリが必要である。ノード500¹⁷に対応する、データフィールド406中のビット502¹⁷は‘1’にセットされ、r3を記憶するマップエントリ504⁹がサブツリーマップ418(図5)に加えられる。

【0056】

ノード500¹⁸ - 500³²はすべてノード500¹⁷と同じルートを共有し、サブツリーマップ418(図5)で新たなマップエントリは必要ない。対応するビット502¹⁸ - 502³²は‘0’にセットされる。したがって、32個のノード500¹ - 500³²について、ルートエントリ302(図4)またはサブツリーエントリ304(図4)を記憶するには、マップエントリ504¹⁻⁹が9つ必要である。

40

【0057】

ノード500¹ - 500³²に対応するマップエントリ504¹ - 504⁹は、データフィールド406に記憶される稠密サブツリー記述子に記憶される‘1’の数を計算することにより、サブツリーマップ418(図5)中でインデックス付けされる。たとえば、ノード500²⁸に対応するマップエントリ504¹ - 504⁹を見出すには、データフィールド406のビット502¹ - 502²⁸に記憶される‘1’の数をカウントする。‘1’の数が8であると、対応するマップエントリはデフォルトルートから8番目の場所、すなわち、マップエントリ504⁹である。

50

【 0 0 5 8 】

ルート変更がある場合にのみマップエントリを記憶することにより、サブツリーマップ 4 1 8 (図 5) 中のサブツリー当りのマップエントリ $504^1 - 504^9$ の数が減少する。

【 0 0 5 9 】

図 7 は図 5 に示されたポインタフィールド 4 0 8 を図示する。ポインタフィールド 4 0 8 は、サブツリーマップ 4 1 8 (図 5) 中の連続する 1 6 のマップエントリ場所 $504^1 - 504^{16}$ (図 6 B) のブロックのベースアドレスを記憶するためのブロックベースアドレスフィールド 600^1 、 600^2 を含む。連続する 1 6 のマップエントリのブロック 602^1 、 602^2 中のサブツリーマップ 4 1 8 (図 5) にメモリが割当てられる。8 レベルのサブツリーは異なるルートを 2 5 6 まで有することができ、2 5 6 のすべてのルートを記憶するためには、ブロック 602^1 、 602^2 が 1 6 個必要である。必要なブロック 602 の数は、サブツリーに対するルートの数に依存する。ブロックベースアドレス 602^1 、 602^2 をブロックベースアドレス (図示せず) のフリーリストから除くことにより、ブロック 602 を特定のサブツリーに割当てる。メモリにアドレスのフリーリストを与えるための方法は技術分野では周知である。

10

【 0 0 6 0 】

1 6 のマップエントリ 504^{1-16} のメモリブロックを割当てることにより、サブツリーマップ 4 1 8 (図 5) 中のメモリの扱いがより容易になる。なぜなら割当てられた 1 6 個の場所は連続しているからである。

【 0 0 6 1 】

20

図 8 は、図 5 に示されたマップアドレス論理 4 0 2 を図示する。マップアドレス論理 4 0 2 は、オフセット論理 7 0 0、ポインタ論理 7 0 2 およびアダー論理 7 0 4 を含む。オフセット論理 7 0 0 はノードセレクト論理 7 0 6 および 1 カウント論理 7 0 8 を含む。ポインタ論理はベースアドレスセレクト論理 7 1 0 を含む。

【 0 0 6 2 】

ノードセレクト論理 7 0 6 は、キー 2 1 0 b の 8 ビットに対応する、サブツリーデータ 4 1 2 中のノード 5 0 0 (図 6 B) を選択する。対応するノード番号はノードセレクト 7 1 8 上を 1 カウント論理 7 0 8 に転送される。1 カウント論理 7 0 8 は、サブツリーデータフィールド 4 0 6 に記憶された ' 1 ' の数を、選択されたノード 5 0 0 に対応するビットまでカウントする。1 の数はブロックセレクト 7 1 2 上をポインタ論理 7 0 2 へおよびブロックオフセット 7 1 4 上をアダー論理 7 0 4 へ転送される。

30

【 0 0 6 3 】

8 ビット幅のカウントフィールドを必要とする 2 5 6 ビットのサブツリーデータフィールド 4 0 6 に記憶される ' 1 ' は最大で 2 5 6 個であり得る。8 ビットカウントフィールドは 2 つのフィールドに分割され、4 つの M S B がブロックセレクト 7 1 2 を与え、4 つの最下位ビット (L S B) がブロックオフセット 7 1 4 を与える。

【 0 0 6 4 】

たとえば、8 ビットキー 2 1 0 b が ' 0 1 0 0 0 1 0 0 ' である場合、ノード番号 6 8 を選択し、サブツリーデータ 4 1 2 の最初の 6 8 ビットに記憶される ' 1 ' が 2 7 個存在すれば、カウントは 1 C H e x (0 0 0 1 1 1 0 0)、M S B (0 0 0 1) である。すなわち、ブロックセレクト 7 1 4、セレクトブロック 602^1 (図 6) および L S B (1 1 0 0) である。すなわち、ベースブロックオフセットセレクトマップエントリ 504^{11} (図 6) であり、すなわちブロック 502^1 中の 1 2 番目のエントリである。

40

【 0 0 6 5 】

ベースアドレスセレクト論理 7 1 0 は、オフセット論理 7 0 0 から転送されるブロックセレクト 7 1 2 に依存して、サブツリーポインタ 4 1 4 からベースアドレス 7 1 6 を選択する。アダー論理 7 0 4 は、オフセット論理 7 0 0 から転送されるブロックオフセット 7 1 4 をベースアドレス 7 1 6 に加算し、マップアドレス 4 1 6 を与える。マップアドレス 4 1 6 はマップ 1 0 6 b - d 中のマップエントリ 504 (図 6 B) のインデックスである。

【 0 0 6 6 】

50

図 9 は、この発明の原則に従う、ルックアップテーブル 200 中のキー 210 (図 3) に対して最長一致プレフィックスをサーチするためのステップを図示するフローチャートである。

【0067】

ステップ 800 で、キー 210 a (図 3) の最初の部分がマップ 106 a へのインデックスとして転送される。処理はステップ 802 に継続する。

【0068】

ステップ 802 で、キー 210 a (図 3) の最初の部分によってインデックス付けされた第 1 のレベルのマップ中のマップエントリ 504 (図 6 B) に記憶されたマップエントリデータ 220 a (図 3) は、次のレベルのさらなるサーチが必要であるか否かを判定する。必要ならば、処理はステップ 804 に継続する。必要なければ、第 1 のレベルのマップ中のインデックス付けされたマップエントリ 504 (図 6 B) 中のルートエントリ 302 (図 4) が、キーに対して、対応する最長プレフィックスルートを記憶し、処理はステップ 808 に継続する。

【0069】

ステップ 804 で、次のレベルのマップ 106 b - d がサーチされる。次のレベルのマップのためのインデックスは、前のレベルのマップ中のインデックス付けされたマップエントリ 504 (図 6 B) 中のサブツリーエントリ記述子 304 (図 4) に記憶されたサブツリーインデックス 312 と、キー 210 b - d の次の部分とに依存する。処理はステップ 806 に継続する。

【0070】

ステップ 806 で、次のレベルのマップ中のインデックス付けされたマップエントリ 504 (図 6 B) は、キーに対する、対応する最長プレフィックスルートインデックスまたはさらなるサーチが必要であることを示すサブツリーインデックスを記憶する。さらなるサーチが必要ならば、処理はステップ 804 に継続する。必要なければ、処理はステップ 808 に継続する。

【0071】

ステップ 808 で、マップ 106 a - d のうち 1 つの中のマップエントリ 504 (図 6 B) に記憶されたルートインデックス 102 (図 3) は、ルートインデックス 102 (図 3) としてルックアップテーブル 200 から転送される。処理は完了する。

【0072】

深度拡張 (depth expansion)

図 3 に示されたルックアップテーブル 200 に記憶可能なルートインデックス 102 (図 3) の数は、サブツリーマップ 418 (図 5) 中の利用可能なマップエントリ 504 (図 6 B) の数によって限定される。たとえば、各々のサブツリーマップ 418 (図 5) が 128 K のマップエントリを含みかつルックアップテーブル中にサブツリーマップ 418 (図 5) が 2 つ存在すれば、ルックアップテーブル 200 中に最大で 256 K のルートインデックス 102 (図 3) を記憶可能である。128 K のマップエントリを有するサブツリーマップ 418 (図 5) は 17 ビットのインデックスを必要とする。512 K のマップエントリを有するサブツリーマップ 418 (図 5) は 19 ビットのインデックスを必要とする。ルックアップテーブル 200 中の 2 つの 512 K サブツリーマップ 418 (図 5) は、32 ビット IPv4 デスティネーションアドレスのために、可能な 40 億個のルートインデックス 102 (図 3) のうち 100 万個のための記憶を提供する。

【0073】

ルートインデックス 102 (図 3) を記憶するためのマップエントリ 504 (図 6 B) の数は、複数のルックアップテーブル 200 を設けることによって増やすことができる。複数のルックアップテーブルは、ルックアップテーブル 200 のうち 1 つの中のサブツリーマップ 418 (図 5) のマップエントリ 504 (図 6 B) に記憶された、サーチキー 210 に対応する値を求めて並行してサーチされる。

【0074】

10

20

30

40

50

図10Aは深度拡張のための実施例を図示する。2つのルックアップテーブル、すなわちマスタルックアップテーブル200aおよびスレーブルックアップテーブル200bが示される。しかしながら、ルックアップテーブルの数は示された2つに限定されるのではなく、1つよりも多くのスレーブルックアップテーブル200bを加えることができる。

【0075】

ルックアップテーブル200a-bの各々は、同じサーチキー210によって並行してサーチされる。サーチキー210に対応するルートインデックス102(図3)は、ルックアップテーブル200a-bのうち1つの中のサブツリーマップ418(図5)またはルックアップテーブル200a、200bの両者の中のダイレクトマップ106aに記憶される。最終ルートインデックス900は、両者のルックアップテーブル200a、200bの並行したサーチの後に見出される。

10

【0076】

図10Bは、図10Aに示された実施例のルックアップテーブル200aの1つを図示する。ルックアップテーブル200a-bの各々は、図3と関連してルックアップテーブル200について既に説明されたようなマップ106a-dおよびパイプライン208ならびにドライバ論理902を含む。ルックアップテーブル200aは、サーチキーに対応するルートインデックスを求めて、マップ106a-d中の複数レベルサーチを行なう。各レベルのサーチ結果は、マップエントリデータ220a-d上をパイプライン208に転送される。パイプライン208は、サーチ904の結果をドライバ論理902に転送する。ルックアップテーブル200a-bの各々の中のマップ106aのマップエントリ504(図6B)はサブツリーエントリ304(図4)を記憶するが、ルートエントリ302(図4)は、マスタルックアップテーブル200a中のマップ106aにしか記憶されない。ルートエントリ302の代わりに、スレーブルックアップテーブル200b中のマップ106aのマップエントリ504(図6B)にノーエントリ300(図4)が記憶される。マップ106a中でただ1つのルックアップテーブルにルートインデックスを配することにより、ルックアップテーブルを選択して最終ルートインデックス900を与えることを回避する。この結果、スレーブルックアップテーブル200b中のメモリ64Kが与えられる。これは、ルートインデックスを記憶するのに用いることはできないが、図3と関連して説明されたように、同じルックアップテーブルをマスタルックアップテーブルとしてまたはスレーブルックアップテーブルとして構成できるようにする。代替的な実施例では、マップ106aを有しないスレーブルックアップデバイスを設けることができる。ルートインデックス102(図3)をマップ106a中のルートエントリ302(図4)に記憶すれば、サーチは、マスタルックアップテーブル200a中のマップ106aで終了する。

20

30

【0077】

図10Aに示されたように、マスタルックアップテーブル200aおよびスレーブルックアップテーブル200bは最終ルートインデックス900を共有する。最終ルートインデックス900を記憶するルックアップテーブル200a、200bはルートインデックス102(図3)を与える。ルックアップテーブル200a、200bの各々が別個のデバイスならば、最終ルートインデックス900を共有することにより、各デバイスの外部ピンのカウントが減少する。いかなるときにも、最終ルートインデックス900を駆動するのは、ルックアップテーブル200a、bのうち1つだけである。

40

【0078】

サーチキー210と関連のルートインデックスをルックアップテーブル200a、200bの両者に記憶して、ルックアップテーブルの両者が同時に最終ルートインデックス900を駆動するエラー状態を回避するため、各ルックアップテーブル200a、bはデバイスコード906を記憶する。3ビットのデバイスコードにより、拡張ルックアップテーブルが8つのデバイスを含むことが可能になる。

【0079】

ドライバ論理902は、サーチ904の結果がルートインデックス102(図3)を含む

50

か否かを判定する。含んでいれば、ルックアップテーブル 200 a 中のドライバ論理 902 は、バスリクエスト信号（図示せず）上に、最終ルートインデックス 900 を駆動する意図を信号で伝える。2 つ以上のルックアップテーブル 200 a、200 b が同時にルートインデックス信号を駆動する意図を信号で伝えれば、ルートインデックスは、ルックアップテーブル 200 a、200 b により、最も低いデバイスコードを与えられる。バスリクエスト信号を用いることによりバス競合を解決するための方法は技術分野で周知である。

【0080】

図 10 C は、サーチキー 210 に対応する値を記憶するのに利用可能なマップエントリ数を増やすために深度拡張を与える別の実施例を図示する。図 10 C に示される実施例では、2 つのルックアップテーブル 200 c - d、すなわちマスタルックアップテーブル 200 c およびスレーブルックアップテーブル 200 d が値を記憶するために設けられる。しかしながら、ルックアップテーブルの数は示された 2 つに限定されるものではなく、マップエントリ数は、より多くのスレーブルックアップテーブル 200 d を加えることによって増やすことができる。サーチキー [39:0] 210 に対応するルックアップテーブル 200 c - d のうち 1 つの中のマップエントリに記憶される値に対するサーチは、ルックアップテーブル 200 c - d で並行して行なわれる。

【0081】

図 10 D は、図 10 C に示された実施例のスレーブルックアップテーブル 200 d を図示する。各々のルックアップテーブルは、図 3 と関連してルックアップテーブル 200 について説明されたような、マップ 106 a - d を含む。ルックアップテーブル 200 c - d の各々の中のマップ 106 a のマップエントリは、サブツリーエントリ 304（図 4）を記憶する。各々のルックアップテーブル c - d は、マップ 106 a 中のマップエントリ 504（図 6 B）に記憶されたサブツリーエントリ記述子 304（図 4）に記憶されたサブツリーインデックス 312 を次のマップ 106 b - d に転送する。しかしながら、ルートインデックス 102（図 3）は、マスタルックアップテーブル 200 c 中のマップ 106 a にのみ記憶される。スレーブルックアップテーブル 200 d 中のマップ 106 a にノーエントリを記憶して、キーに対応するルートインデックスを 1 つよりも多くのルックアップテーブル 200 b、200 d に記憶しないようにする。

【0082】

スレーブルックアップ 200 d の複数レベルのサーチ 904 の結果は最終インデックス論理 1004 に転送される。最終インデックス論理 1004 は、複数レベルサーチ 904 の結果およびマスタルックアップテーブル 200 c から転送された入結果（incoming result）1000 a を出結果（outgoing result）1002 a として転送する。ルートインデックス 102（図 3）が複数レベルサーチ 904 の結果に含まれれば、複数レベルサーチの結果が出結果 1002 a として転送される。ルートインデックス 102（図 3）が入結果 1000 a に含まれれば、入結果 1000 a が出結果 1002 a として転送される。ルートインデックス 102（図 3）が入結果 1000 a または複数レベルサーチ 904 の結果のいずれにも含まれなければ、複数レベルサーチ 904 の結果が出結果 1002 a として転送される。

【0083】

図 10 C に示されるように、マスタルックアップテーブル 200 c およびスレーブルックアップテーブル 200 d は、入結果 1000 a で標識付けられた共通のバスを介して接続される。ルートインデックス 102（図 3）は、出結果 1002 a の上を、スレーブルックアップテーブル 200 d から転送されるのみである。1 つよりも多くのスレーブルックアップテーブル 200 d が存在する場合、最後のスレーブルックアップテーブルにより、拡張ルックアップテーブルに対するルートインデックス 102（図 3）が与えられる。この実施例は、図 10 A と関連して説明されたマルチドライバ最終ルートインデックス 900 の実現例を回避するが、入結果 1000 a 用に、より多くのデバイス外部ピンが必要である。

10

20

30

40

50

【 0 0 8 4 】

図 1 1 A - B は、ルックアップテーブル 2 0 0 a - b (図 1 0 A) または 2 0 0 c - d (図 1 0 C) 中の、図 2 B に示されたルートの分布のバイナリツリー表示を図示する。

【 0 0 8 5 】

図 1 1 A は、マスタルックアップテーブル 2 0 0 a (図 1 0 A) または 2 0 0 c (図 1 0 C) に記憶されたルートのバイナリツリー表示を図示する。図 2 B に示されたルートのバイナリツリー表示に示されるサブツリー B は、マスタルックアップテーブル 2 0 0 a には含まれない。ノード $130^1 - 130^{22}$ および 130^{24-32} は、図 3 と関連して説明されたように、ルックアップテーブル 2 0 0 a 中のマップ 1 0 6 a でコード化される。サブツリー B がインデックス付けされるノードは、それがマスタルックアップテーブル 2 0 0 a に記憶されれば、X でグラフィックに表わされ、ブルーニングされたサブツリーを示す。マスタルックアップテーブル 2 0 0 a 中のノード 130^{23} に対応するマップエントリ 5 0 4 (図 6 B) は、もはやサブツリー B へのサブツリーインデックス 3 1 2 (図 4) を記憶しない。代わりに、ノード 130^{23} に対応するマップエントリが別のスレーブルックアップテーブル 2 0 0 b 中のサブツリーマップ 4 1 8 (図 5) に記憶されることを示すマスタルックアップテーブル 2 0 0 a 中の、ノード 130^{23} に対応するマップエントリ 5 0 4 (図 6 B) に、ノーエントリ 3 0 0 (図 4) が記憶される。

10

【 0 0 8 6 】

図 1 1 B は、スレーブルックアップテーブル 2 0 0 b (図 1 0 A) または 2 0 0 d (図 1 0 C) 中のサブツリーマップ 4 1 8 (図 5) 中のマップエントリ 5 0 4 (図 6 B) に記憶されるルートのバイナリツリー表示を図示する。スレーブルックアップテーブル 2 0 0 b に記憶されるルートのバイナリツリー表示は、サブツリー A が含まれないという点で、図 2 B に示されたバイナリツリー表示と異なっている。したがって、図 2 B と関連して説明されたように、ノード $130^1 - 130^3$ および $130^5 - 130^{32}$ がコード化される。スレーブルックアップテーブル 2 0 0 b 中のノード 130^4 に対応するマップエントリ 5 0 4 (図 6 B) は、もはやサブツリー A へのサブツリーインデックス 3 1 2 (図 4) を記憶しない。代わりに、スレーブルックアップテーブル 2 0 0 b 中のノード 130^4 に対応するマップエントリは、ノード 130^4 に対応するマップエントリが別のルックアップテーブルに記憶されることを示すノーエントリ 3 0 0 (図 4) を記憶する。サブツリー A へのサブツリーインデックスおよびしたがってホスト 1 3 8 (図 1 1 A) へのルートインデックスはマスタルックアップテーブル 2 0 0 a に記憶され、サブツリー B へのサブツリーインデックスおよびしたがってホスト 1 4 0 へのルートインデックスはスレーブルックアップテーブル 2 0 0 b に記憶される。スレーブルックアップテーブル 2 0 0 b、2 0 0 d はサブツリーに対する結果のみを記憶する。すなわち、スレーブルックアップテーブル 2 0 0 b、2 0 0 d は第 1 のレベルのマップ 1 0 6 a の結果を記憶しない。

20

30

【 0 0 8 7 】

図 1 1 A および図 1 1 B を参照して、キー 2 1 0 a の最初の部分による、スレーブマップレベル 1 1 1 0 4 a またはマスタマップレベル 1 1 1 0 2 (図 3) 中のノード $130^9 - 130^{12}$ のいずれに対するサーチの結果も、マスタルックアップテーブル 2 0 0 a、2 0 0 c 中のマップ 1 0 6 a のマップエントリ 5 0 4 (図 6 B) 中のルートエントリ 3 0 2 (図 4) と、スレーブルックアップテーブル 2 0 0 b、2 0 0 d 中のマップ 1 0 6 a のマップエントリ 5 0 4 (図 6 B) に記憶されるノーエントリ 3 0 0 (図 4) とに記憶される r 1 1 1 6 となる。マスタルックアップテーブル 2 0 0 a、2 0 0 c に記憶されるルートエントリ 3 0 2 (図 4) は、入結果 1 0 0 0 a 上をスレーブルックアップテーブル 2 0 0 b、2 0 0 d に転送され、スレーブルックアップテーブル 2 0 0 b、2 0 0 d により出結果 1 0 0 2 a 上を転送される。

40

【 0 0 8 8 】

キー 2 1 0 a の最初の部分による、ノード 130^4 に対するサーチの結果は、マスタルックアップテーブル 2 0 0 a 中のマップ 1 0 6 a のマップエントリ 5 0 4 (図 6 B) 中にサブツリーエントリ記述子 3 0 4 (図 4) に記憶されたサブツリー A に対するサブツリーイ

50

ンデックス 3 1 2 (図 4) となる。サブツリーインデックス 3 1 2 はマスタルックアップテーブル 2 0 0 a 中のマップ 1 0 6 b に転送され、マスタルックアップテーブル 2 0 0 a に記憶されるルートエントリ 3 0 2 (図 4) に対するサーチを継続する。

【 0 0 8 9 】

キー 2 1 0 の最初の部分による、ノード 1 3 0²³ に対するサーチの結果は、マスタルックアップテーブル 2 0 0 a 中のマップ 1 0 6 a のマップエントリ 5 0 4 (図 6 B) に記憶されるサブツリーエントリ記述子 3 0 4 (図 4) と、スレーブルックアップテーブル 2 0 0 b 中のマップ 1 0 6 a のマップエントリ 5 0 4 (図 6 B) に記憶されるノードエントリ 3 0 0 (図 4) とである。したがって、ルートエントリ 3 0 2 (図 4) に対するサーチは、スレーブルックアップテーブル 2 0 0 b 中のマップ 1 0 6 b で、キー 2 1 0 b の次の部分を用いて継続する。

10

【 0 0 9 0 】

図 1 2 は、図 1 0 A に示されたルックアップテーブル 2 0 0 a - b 中のマップエントリ 5 0 4 (図 6 B) に記憶されるべきルートエントリ 3 0 2 (図 4) を分散するための方法を図示するフローチャートである。同じ方法は、図 1 0 C に示されたルックアップテーブル 2 0 0 c - d に当てはまる。マップエントリに記憶されるべきルートエントリ 3 0 2 (図 4) はまず、ルックアップテーブル 2 0 0 a - b に記憶される前に、プロセッサ (図示せず) によってメモリに記憶される。

【 0 0 9 1 】

ルートエントリ 3 0 2 (図 4) がメモリに記憶される間、ルックアップテーブル 2 0 0 a - b (図 1 0 A) の各々に記憶されるべきルートエントリ 3 0 2 (図 4) の数がカウントされる。マップレベル 1 1 1 0 4 a (図 1 1 B) に対するルートエントリ 3 0 2 (図 4) は、ルックアップテーブル 2 0 0 a 中のマップ 1 0 6 a に記憶される。マップレベル 1 1 1 0 4 a (図 1 1 B) に対するサブツリーエントリ (図 4) は、ルックアップテーブル 2 0 0 a - 2 0 0 b の各々の中のマップ 1 0 6 a に記憶される。

20

【 0 0 9 2 】

ステップ 1 2 0 0 で、ルックアップテーブル 2 0 0 a - 2 0 0 b (図 1 0 A) の各々の中のマップ 1 0 6 a のサブツリーエントリ 3 0 4 (図 4) の各々ごとの、記憶されるべきルートエントリ 3 0 2 (図 4) の数を計算して、どのようにルートエントリ 3 0 2 (図 4) をルックアップテーブル 2 0 0 a - b (図 1 0 A) の中に分散すべきかを判定する。ルートエントリ 3 0 2 (図 4) を記憶するのに必要なマップエントリ 5 0 4 (図 6 B) の総数を判定した後、処理はステップ 1 2 0 2 に継続する。

30

【 0 0 9 3 】

ステップ 1 2 0 2 で、サブツリーに対して記憶されるべきマップエントリ 5 0 4 (図 6 B) の総数を、ルックアップテーブル 2 0 0 a - b (図 1 0 A) の数で除算して、各ルックアップテーブル 2 0 0 a - b (図 1 0 A) に記憶すべきルートエントリ 3 0 2 (図 4) の数を定める。処理はステップ 1 2 0 4 に継続する。

【 0 0 9 4 】

ステップ 1 2 0 4 で、ルートエントリ 3 0 2 (図 4) は、選択されたルックアップテーブル 2 0 0 a - b 中のサブツリーマップ 4 1 8 (図 5) 中のマップエントリ 5 0 4 (図 6 B) に記憶される。処理はステップ 1 2 0 6 に継続する。

40

【 0 0 9 5 】

ステップ 1 2 0 6 で、選択されたルックアップテーブル 2 0 0 a - b (図 1 0 A) 中のサブツリーマップ 4 1 8 (図 5) のマップエントリ 5 0 4 (図 6 B) に記憶されたルートエントリ 3 0 2 の数が 1 / n 未満ならば、なおここで n は利用可能なルックアップテーブル 2 0 0 a - b (図 1 0 A) の数であるが、プロセスはステップ 1 2 0 4 に続く。それ以上ならば、選択されたルックアップテーブル 2 0 0 a - b はマップエントリの総数の 1 / n を記憶しており、処理はステップ 1 2 0 8 に継続する。

【 0 0 9 6 】

ステップ 1 2 0 8 で、選択されたルックアップテーブル 2 0 0 a - b はマップエントリの

50

総数の $1/n$ を記憶し、選択されたルックアップテーブル 200 a - b のいずれの残余のサブツリーノードに対しても、ノーエントリ 300 (図4) を記憶する。それぞれのサブツリーに対するルートインデックスは、現在選択されたルックアップテーブルに記憶されないからである。処理はステップ 1210 に続く。

【0097】

ステップ 1210 で、すべてのルートエントリが記憶されれば、処理は完了する。記憶されていないければ、処理はステップ 1212 に継続する。

【0098】

ステップ 1212 で、次のルックアップテーブル 200 a - b (図10A) が選択される。処理はステップ 1204 に続く。

10

【0099】

ルートエントリは、IPアドレスに対応するルートインデックスに対するサーチの前に、ルックアップテーブル 200 a - b (図10A) の中に分散される。サーチは、ルックアップテーブル 200 a - b (図10A) の各々の中で並行して行なわれる。ルックアップテーブルの各々の中で並行して実現されるサーチのための方法が、ルックアップテーブル 200 a - b (図10A) の1つのために説明される。

【0100】

図13は、図10Cに示されたルックアップテーブル 200 c - d のいずれの1つにも記憶されるサーチキーに対応する値に対する、サーチキーによるサーチのための方法を図示するフローチャートである。

20

【0101】

ステップ 1300 で、ルックアップテーブル 200 c - d (図10C) の各々はサーチキー 210 を受ける。ルックアップテーブル 200 c - d の各々の中のマップ 106 a は、キー 210 a の最初の部分に対応する値を求めてサーチされる。処理はステップ 1302 に継続する。

【0102】

ステップ 1302 において、マップ 106 a 内のマップエントリ 504 (図6B) に記憶されたエントリが読出される。マスタルックアップテーブル 200 c 中のマップエントリ 504 (図6B) は、ノーエントリ 300 (図4)、ルートエントリ 302 (図4) またはサブツリーエントリ記述子 304 (図4) を記憶し得る。スレーブルックアップテーブル 200 d 内のマップエントリ 504 (図6B) は、ノーエントリ 300 (図4) およびサブツリーエントリ記述子 304 (図4) を記憶し得る。それぞれのルックアップテーブル 200 内のマップエントリがルートエントリ 302 (図4) を記憶していれば、そのエントリは有効な値であり、ルックアップテーブル 200 c - 200 d 内の後続のマップ 106 b - d のさらなるサーチは必要はなく、処理はステップ 1310 へ進む。そうでなければ、処理はステップ 1304 へ進む。

30

【0103】

ステップ 1304 において、このエントリがサブツリーエントリ記述子 304 (図4) を記憶していれば、ルックアップテーブル 200 c - d 内のさらなるサーチが必要となり、処理はステップ 1306 へ進む。そうでなければ、このエントリはノーエントリを記憶しており、これがさらなるサーチが必要でないことを示しているので、処理はステップ 1310 へ進む。

40

【0104】

ステップ 1306 において、選択されたサブツリーにおいてサーチが継続される。キー 210 b - d の次の部分および前のレベルのサーチの結果得られたサブツリーインデックス 312 (図4) に依存して、次のレベルのマップ 106 b - d (図3) がサーチされる。処理はステップ 1308 へ進む。

【0105】

ステップ 1308 において、現在のレベルのマップ 106 b - d のサーチの結果得られるマップエントリに依存して、サーチを継続するか否かが判定される。マップエントリ 50

50

4 (図 6 B) がサブツリーエントリ記述子 3 0 4 (図 4) を記憶していれば、次のレベルのマッパ 1 0 6 b - d でサーチが継続され、処理はステップ 1 3 0 6 へ進む。マッパエントリ 5 0 4 (図 6 B) がサブツリーエントリ記述子 3 0 4 (図 4) を記憶していなければ、さらなるサーチは必要なく、処理はステップ 1 3 1 0 へ進む。

【 0 1 0 6 】

ステップ 1 3 1 0 において、サーチの結果が別のルックアップテーブルから受けた入結果 1 0 0 0 a と比較される。たとえば、ルックアップテーブルがスレーブルックアップテーブル 2 0 0 d の場合、マスタルックアップテーブル 2 0 0 c でのサーチからの入結果は、入結果 1 0 0 0 a 上でルックアップテーブル 2 0 0 d へ転送され、スレーブルックアップテーブル 2 0 0 d におけるサーチの結果と比較される。処理はステップ 1 3 1 2 へ進む。

10

【 0 1 0 7 】

ステップ 1 3 1 2 において、入結果 1 0 0 0 a と現在のルックアップテーブル 2 0 0 d のサーチ結果が異なる場合、処理はステップ 1 3 1 4 へ進む。入結果 1 0 0 0 a と現在のルックアップテーブル 2 0 0 d のサーチ結果との両者が同じであれば、別々のルックアップテーブル 2 0 0 c - d 中のマッパエントリ 5 0 4 (図 6 B) に 2 つの有効な結果が記憶されている。同じキー 2 1 0 に対して 2 つの有効な結果を記憶すべきでなく、処理はステップ 1 3 1 6 へ進む。

【 0 1 0 8 】

ステップ 1 3 1 4 において、入結果 1 0 0 0 a がチェックされ、それが有効であるか否かが判定される。入結果 1 0 0 0 a は、それがルートエントリ 3 0 2 (図 4) であれば有効である。入結果 1 0 0 0 a は、それがノーエントリ 3 0 0 (図 4) またはサブツリーエントリ記述子 3 0 4 (図 4) であれば無効である。サブツリーエントリ記述子 3 0 4 (図 4)、ルートエントリ 3 0 2 (図 4) およびノーエントリ 3 0 0 (図 4) については図 4 に関連して既に説明した。入結果 1 0 0 0 a が無効であれば、処理はステップ 1 3 1 8 へ進む。そうでなければ、処理はステップ 1 3 2 0 へ進む。

20

【 0 1 0 9 】

ステップ 1 3 1 8 において、入結果 1 0 0 0 a は有効であり、現在のルックアップテーブル 2 0 0 d のサーチ結果は無効である。入結果 1 0 0 0 a は現在のルックアップテーブル 2 0 0 d から出結果 (出て行く結果) 1 0 0 2 a 上で転送される。入結果 1 0 0 0 a は、現在のルックアップテーブル 2 0 0 d が最後のルックアップテーブルであればルートインデックス 1 0 2 (図 3) として転送され、または、入結果 1 0 0 0 a として次のルックアップテーブルへ転送される。処理が完了する。

30

【 0 1 1 0 】

ステップ 1 3 1 6 において、2 つの有効な結果の値がキーに対して別々のルックアップテーブルに記憶される。ルックアップテーブル 2 0 0 c - d 中のルートエントリの記憶中にエラーが起こる。エラーコードが生成されるのでこのエラーは修正され得る。処理が完了する。

【 0 1 1 1 】

ステップ 1 3 2 0 において、現在のルックアップテーブル 2 0 0 d のサーチ結果も入結果 1 0 0 0 a も有効でない。現在のルックアップテーブル 2 0 0 d のサーチ結果は、たとえ無効であっても、入結果 1 0 0 0 a として次のルックアップテーブルへ転送される。処理が完了する。

40

【 0 1 1 2 】

図 1 4 は、図 1 0 A に示すルックアップテーブル 2 0 0 a - d のうちの 1 つに記憶されたサーチキーに対応する値をサーチするための方法を表わすフローチャートである。

【 0 1 1 3 】

ステップ 1 3 4 0 において、ルックアップテーブル 2 0 0 a および 2 0 0 b の両者における第 1 のレベルのマッパ 1 0 6 a が、キー 2 1 0 a の第 1 の部分に対応する値についてサーチされる。処理はステップ 1 3 4 2 へ進む。

【 0 1 1 4 】

50

ステップ 1 3 4 2 において、キー 2 1 0 a の第 1 の部分で第 1 のレベルのマッパ 1 0 6 a をサーチした後に有効な結果の値が見つければ、処理はステップ 1 3 5 2 へ進む。そうでなければ、処理はステップ 1 3 4 4 へ進む。

【 0 1 1 5 】

ステップ 1 3 4 4 において、キー 2 1 0 a の第 1 の部分で第 1 のレベルのマッパ 1 0 6 a をサーチした結果の値がサブツリーエントリ記述子 3 0 4 (図 4) であれば、処理はステップ 1 3 4 6 へ進む。そうでなければ、そのキーに対する有効な値は現在のルックアップテーブルには記憶されておらず、処理は完了する。

【 0 1 1 6 】

ステップ 1 3 4 6 において、有効な値のサーチは、前のレベルのマッパのサーチ中に見つかったサブツリーエントリ記述子 3 0 4 (図 4) において識別されたサブツリーで継続される。次のレベルのマッパが、キー 2 1 0 b - c の次の部分および次のレベルのサーチの結果得られるサブツリーセレクトに依存する値に対してサーチされる。処理はステップ 1 3 4 8 へ進む。

【 0 1 1 7 】

ステップ 1 3 4 8 において、このサーチ結果により、次のレベルのマッパのサーチが要求されるか否かが判定される。現在のサーチの結果得られるエントリは、ルートエントリ 3 0 2、ノーエントリ 3 0 0 (図 4) またはサブツリーエントリ記述子 3 0 4 (図 4) を記憶し得る。このエントリがサブツリーエントリ記述子 3 0 4 (図 4) を記憶していれば、さらなるサーチが必要であり、処理はステップ 1 3 4 6 へ進む。このエントリがサブツリーエントリ記述子 3 0 4 (図 4) を記憶していなければ、処理はステップ 1 3 5 0 へ進む。

【 0 1 1 8 】

ステップ 1 3 5 0 において、このエントリがルートインデックス 1 0 2 (図 3) を記憶していれば、処理はステップ 1 3 5 2 へ進む。そうでなければ、このエントリは別のルックアップテーブルに記憶されている。処理が完了する。

【 0 1 1 9 】

ステップ 1 3 5 2 において、キーに対応する有効な値が現在のルックアップテーブルに記憶されている。この有効な値は、キーに対応するルートインデックス 1 0 2 (図 3) として転送される。処理が完了する。

スパスモード

図 5 を参照して、サブツリーエントリ 4 0 4 は 2 5 6 までの可能なルートインデックスのアクセスを、2 5 6 のノードサブツリーにおいて各ノードに 1 つずつ提供する。これらのルートインデックスはサブツリーマッパ 4 1 8 (図 5) におけるマッパエントリ 5 0 4¹ - 5 0 4ⁿ に記憶される。サブツリーマッパ 4 1 8 (図 5) 内のマッパエントリ 5 0 4 (図 6 B) に対するマッパアドレス 4 1 6 は、データフィールド 4 0 6 に記憶された稠密サブツリー記述子と、サブツリーエントリ 4 0 4 内のポインタフィールド 4 0 8 に記憶されたサブツリーポインタとに依存して決定される。稠密サブツリー記述子の形式については図 6 A - 6 B に関連して既に説明した。稠密サブツリー記述子は、2 5 6 のノードサブツリーにおける各ノードに対してノードビット 5 0 2 (図 6 B) を記憶する。しかしながら、すべてのサブツリーが、2 5 6 個のノードの各々に対してある別個のルートインデックスを有する。たとえば、1 つのサブツリーは 1 つのルートインデックスのみを有し得る。

【 0 1 2 0 】

図 1 5 は、第 1 のマッパレベル 1 1 2 a におけるサブツリーエントリ 3 0 4 (図 4) によってインデックス付けされた第 2 のマッパレベル 1 1 2 b における疎サブツリー A および稠密サブツリー B のバイナリツリーを表わした図である。マッパ 1 0 6 a 内の s 1 に対するサブツリーエントリ記述子 3 0 4 (図 4) はサブツリー A のサブツリーエントリ 4 0 4 に対するサブツリーインデックス 3 1 2 を記憶する。マッパ 1 0 6 a 内の s 0 に対するサブツリーエントリ記述子 3 0 4 (図 4) は、サブツリー B のサブツリーエントリ 4 0 4 に対するサブツリーインデックス 3 1 2 を記憶する。

【 0 1 2 1 】

密に配置されたサブツリー B は 1 1 個のルートインデックス、すなわち r_6 から r_{16} と、6 個のサブツリーエントリ、すなわち s_2 から s_7 を有する。サブツリー B に対してルートエントリ 3 0 2 (図 4) とサブツリーエントリ 3 0 4 (図 4) とを記憶するマップエントリ 5 0 4 (図 6 B) に対応するマップアドレス 4 1 6 は、図 6 B に関連して既に述べたように、稠密サブツリー記述子においてコード化される。

【 0 1 2 2 】

疎に配置されたサブツリー A は 2 個のルートインデックス、すなわち r_1 および r_2 を記憶する。これらが稠密サブツリー記述子に記憶されていれば、サブツリーエントリ 4 0 4 全体を用いてマップエントリ 5 0 4 (図 6 B) に対する 3 つのマップアドレス 4 1 6、すなわち r_0 、 r_1 および r_2 が提供される。

10

【 0 1 2 3 】

ルックアップテーブル 2 0 0 に記憶されたルート数は、複数の疎サブツリー記述子のうちの 1 つにある疎サブツリーをコード化し、サブツリーエントリ 4 0 4 内の稠密サブツリー記述子にある密に配置されたサブツリーをコード化することによって、増加し得る。

【 0 1 2 4 】

密に配置されたサブツリーは 1 6 以上のマップエントリ 5 0 4 (図 6 B) を有し、サブツリーエントリ 4 0 4 内のデータフィールド 4 0 6 は、図 6 A - 6 B に関して述べたように、稠密サブツリー記述子を記憶する。疎に配置されたサブツリーは 1 5 以下のマップエントリ 5 0 4 (図 6 B) を有し、サブツリーエントリ 4 0 4 内のデータフィールドは複数の疎サブツリー記述子を記憶する。疎に配置されたサブツリーを疎サブツリー記述子に記憶する能力を与えることにより、サブツリーメモリ 4 0 0 により多くのサブツリーを記憶することができ、よって、ルックアップテーブル 2 0 0 中により多くのルートエントリを記憶することができる。

20

【 0 1 2 5 】

図 1 6 A - C は、図 5 に示すサブツリーエントリ 4 0 4 内のデータフィールド 4 0 6 およびポインタフィールド 4 0 8 と、図 4 に示すサブツリーエントリ記述子 3 0 4 (図 4) とを、サブツリーエントリ 4 0 4 内に複数の疎サブツリー記述子を記憶できるよう変形した例を表わす。

【 0 1 2 6 】

図 1 6 A を参照して、スパースモードで構成されたサブツリーエントリ 4 0 4 内のデータフィールド 4 0 6 は、図 6 B に関連して述べたサブツリーの各ノードごとに 1 ビットを有する稠密サブツリー記述子の代わりに、複数の疎サブツリー記述子 $1400^1 - 1400^n$ を含む。疎サブツリー記述子 $1400^1 - 1400^n$ の各々は、ノード記述子 $1402^1 - 1402^n$ を含む。ノード記述子 $1402^1 - 1402^n$ は、サブツリーにおいて完全に符号化されたルートを表わす 9 ビットの値である。ノード記述子 $1402^1 - 1402^n$ は、サブツリー内の単一のノードまたは複数のノードを表わす。

30

【 0 1 2 7 】

図 1 6 B を参照して、疎サブツリー記述子の記憶をサポートするために、モードフィールド 1 4 0 4 がサブツリーエントリ 4 0 4 内のポインタフィールド 4 0 8 に加えられる。ポインタフィールド 4 0 8 は、ブロックベースアドレス 600^1 およびブロックベースアドレス 600^2 も記憶しており、各ブロックが 1 6 個の割当てられたマップアドレス 4 1 6 を含み、これらのアドレスは各サブツリーエントリ 4 0 4 ごとに合計 3 2 のマップアドレス 4 1 6 を提供する。モードフィールド 1 4 0 4 はモード値を記憶する。モードフィールド 1 4 0 4 に記憶されたモード値は、サブツリーエントリ 4 0 4 に記憶された疎サブツリー記述子 $1400^1 - 1400^n$ の数と、疎サブツリー記述子 $1400^1 - 1400^n$ の各々に記憶されたノード記述子 $1402^1 - 1402^n$ の数とを示す。表 2 は、各モードについてのサブツリーエントリ 4 0 4 の構成を表わす。

40

【 0 1 2 8 】

【 表 2 】

50

	ノード/ サブツリー		ビット	サブツリー/ サブツリー エントリ	ルート/ サブツリーエントリ		未使用	1サブツリーエントリ に要求される マップエントリ	
モード	最大	最小			最大	最小		最大	最小
0	1	1	9	16	16	16	112	32	32
1	2	2	18	10	20	20	76	30	30
2	3	3	27	8	24	24	40	32	32
3	4	4	36	6	24	24	40	30	30
4	7	5	63	4	28	20	4	32	24
5	15	8	135	2	30	16	-14	32	18

表 2

【 0 1 2 9 】

表 2 を参照して、たとえば、サブツリーエントリ 4 0 4 内のポインタフィールド 4 0 8 におけるモードフィールド 1 4 0 4 に記憶されたモード値が「4」にセットされる場合、サブツリーエントリ 4 0 4 の疎サブツリー記述子 1 4 0 0 の各々は 5 から 7 のノード記述子 $1\ 4\ 0\ 2^1 - 1\ 4\ 0\ 2^n$ を記憶する。各ノード記述子 $1\ 4\ 0\ 2^1 - 1\ 4\ 0\ 2^n$ は 9 ビットを記憶する。疎サブツリー記述子 1 4 0 0 に記憶されたビットの総数は、疎サブツリー記述子 $1\ 4\ 0\ 0^1 - 1\ 4\ 0\ 0^n$ ごとのノード記述子 $1\ 4\ 0\ 2^1 - 1\ 4\ 0\ 2^n$ の数を 9 (ノード記述子 $1\ 4\ 0\ 2^1 - 1\ 4\ 0\ 2^n$ ごとのビット数) で乗じることにより計算される。モード 4 に対して疎サブツリー記述子 1 4 0 0 ごとのビット数を計算すると、7 個のノード記述子 1 4 0 2 を有する疎サブツリー記述子 1 4 0 0 は 6 3 ビットを記憶している (7 ノード記述子 * 9 ビット = 6 3)。

【 0 1 3 0 】

サブツリーエントリ 4 0 4 ごとの疎サブツリー記述子 1 4 0 0 の数は、データフィールド 4 0 6 内のビット数を疎サブツリー記述子 $1\ 4\ 0\ 0^1 - 1\ 4\ 0\ 0^n$ 内のビット数で除することにより、算出される。モード 4 について、データフィールド 4 0 6 内のビット数は 2 5 6 であり、疎サブツリー記述子内のビット数は 6 3 である。したがって、疎サブツリー記述子 $1\ 4\ 0\ 0^1 - 1\ 4\ 0\ 0^n$ の数は 4 である ($\text{int}(256 / 63) = 4$)。

【 0 1 3 1 】

サブツリーエントリ 4 0 4 ごとのノード記述子 $1\ 4\ 0\ 2^1 - 1\ 4\ 0\ 2^n$ の総数は、各サブツリーごとのノードの数に各サブツリーエントリ 4 0 4 ごとのサブツリーの数に乗じたものである。モード 4 について計算すると、サブツリーエントリ 4 0 4 ごとのノード記述子 1 4 0 2 の総数は、疎サブツリー記述子 $1\ 4\ 0\ 0^1 - 1\ 4\ 0\ 0^n$ に記憶されたノード記述子 $1\ 4\ 0\ 2^1 - 1\ 4\ 0\ 2^n$ が 7 個ある場合は 2 8 であり ($7 * 4 = 28$)、疎サブツリー記述子 $1\ 4\ 0\ 0^1 - 1\ 4\ 0\ 0^n$ ごとのノード記述子 1 4 0 2 が 5 個ある場合は 2 0 である ($5 * 4 = 20$)。

【 0 1 3 2 】

表 2 のマップエントリの列には、サブツリーマップ 4 1 8 (図 5) 内のマップエントリ 5 0 4 (図 6 B) がいくつサブツリーエントリ 4 0 4 によって使用されているかが示される。マップの値は、サブツリーごとのノードを 1 ずつ増分し、かつ疎サブツリー記述子のサブツリーの数で乗じることにより、算出される。サブツリーのデフォルトエントリを記憶するためには各サブツリーごとのノードの数より 1 以上多いマップエントリが要求される

ので、サブツリーごとのノードは1ずつ増分される。

【0133】

表2のモード4の行を参照して、疎サブツリー記述子1400ごとに7個のノード記述子1402があれば、各サブツリーエントリ404ごとに $32((7+1)*4=32)$ のマッパエントリが要求され、疎サブツリー記述子1400ごとに5個のノード記述子1402があれば、疎サブツリー記述子1400ごとに $24((5+1)*4=24)$ のノード記述子1402が要求される。サブツリーごとのノードの数およびサブツリーエントリ404ごとのサブツリーの数、サブツリーエントリ404ごとのノード記述子の最大数が30を超えないように選択される。これは、サブツリーマッパ418(図5)内のマッパアドレス416が16ブロック刻みで割当てられるからである。ポインタフィールド408に記憶された2つのブロックベースアドレス 600^1 および 600^2 を記憶することにより、32のマッパアドレス416が提供される。

10

【0134】

図16Cを参照して、サブツリーメモリ400における各サブツリーエントリ404は、図6Bに関連して述べたようなデンス(稠密)モードで、またはスパース(疎)モードで構成され得る。デンスモードについて図4に関して述べたサブツリーマッパ418(図5)に記憶されるサブツリーエントリ記述子304(図4)は、サブツリーエントリ404がデンスモードでコード化されるかまたはスパースモードでコード化されるかの指示を提供することによりスパースモードを可能にするよう変更される。この指示(indicator)は、タイプフィールド1406により提供される。

20

【0135】

タイプフィールド1406の状態により、サブツリーエントリ404がデンスモードまたはスパースモードかのいずれで構成されているかが示される。サブツリーエントリ404がスパースモードで構成されている場合、スパースサブツリー記述子セレクトフィールド1408に記憶された値とサブツリーインデックス314とが用いられて、スパースサブツリー記述子1400が選択される。スパースサブツリー記述子セレクト1408については、図16を参照して後により詳細に説明する。

【0136】

図17は、疎に配置されたサブツリーにおけるノードに対してマッパエントリ504(図6B)を選択するようにブロックオフセット714を与えるための、図8に示すオフセット論理700のスパースモード論理1502を表わす。スパースモード論理1502は、サブツリーエントリ404中の疎サブツリー記述子1400に記憶されたノード記述子1402に依存して、ブロックオフセット714を与える。オフセット論理700はまた、デンスモード論理1500も含む。デンスモード論理1500は、密に配置されたサブツリーにおけるルートに対してブロックオフセット714を与えるためのノードセレクト706および1カウント論理708を含む。デンスモード論理1500については、図8に関連して既に説明した。

30

【0137】

タイプフィールド1406の状態がサブツリーエントリ404がスパースモードで構成されていると示す場合、サブツリーエントリ404からのサブツリーデータ412はスパースモード論理1502へ転送される。スパースモードサブツリー論理1502については図18に関連して説明する。

40

【0138】

図18は、図17のオフセット論理700に示されるスパースモード論理1502を表わす。スパースモード論理1502は、サブツリーセレクト論理1600、マルチプレクサ1602、連想メモリ(「CAM」)1606および変換論理1604を含む。選択されたサブツリーエントリ404内のデータフィールド406に記憶された疎サブツリー記述子 $1400^1 - 1400^n$ は、サブツリーデータ412上でオフセット論理700へ転送される。オフセット論理700は疎サブツリー記述子 $1400^1 - 1400^n$ をスパースモード論理1502におけるマルチプレクサ1602へ転送する。サブツリーデータ412中

50

の疎サブツリー記述子 1 4 0 0¹のうちの 1 つが、サブツリーセレクト論理 1 6 0 0 により生成されたセレクト 1 6 1 4 によって選択される。

【 0 1 3 9 】

サブツリーセレクト論理 1 6 0 0 はセレクト 1 6 1 4 を生成し、前のマップレベルにおいて選択されたマップエントリから転送された疎サブツリー記述子セレクト 1 4 0 8 の状態と、選択されたサブツリーエントリ 4 0 4 内のポインタフィールド 4 0 8 に記憶されたモード 1 4 0 4 とに依存して、疎サブツリー記述子 1 4 0 0¹を選択する。表 3 は、モード 4 サブツリーエントリ 4 0 4 について、マルチプレクサ 1 6 0 2 から、選択された疎サブツリー記述子 1 6 1 0 上で転送された、選択された疎サブツリー記述子 1 4 0 0¹およびそれぞれのサブツリーデータビット 4 1 2 を表わす。表 2 のモード 4 の行を参照して、モード 4 サブツリーエントリ 4 0 4 には 4 つの疎サブツリー記述子が記憶され得る。4 つの疎サブツリー記述子 1 4 0 0 の各々は 6 3 ビットであり、7 から 5 個のノード記述子 1 4 0 2 を記憶し得る。よって、これら 4 つの疎サブツリー記述子 1 4 0 0 の各々は 6 3 ビット境界から始まる。第 1 の疎サブツリー記述子 1 4 0 0¹はデータフィールド 4 0 6 のビット 6 2 : 0 に記憶される。第 2 の疎サブツリー記述子 1 4 0 0²はデータフィールド 4 0 6 のビット 1 2 5 : 6 3 に記憶される。第 3 の疎サブツリー記述子 1 4 0 0³はデータフィールド 4 0 6 のビット 1 8 8 : 1 2 6 に記憶され、第 4 の疎サブツリー記述子 1 4 0 0⁴はデータフィールド中のビット 2 5 1 : 1 8 9 に記憶される。データフィールド 4 0 6 におけるそれぞれのビットは、疎サブツリー記述子セレクト 1 4 0 8 により選択される。たとえば、表 3 を見ると、疎サブツリー記述子セレクト 1 4 0 8 が「0 0 0 1」であれば、第 2 の疎サブツリー記述子 1 4 0 0²が選択され、2 5 6 ビットのサブツリーデータ 4 1 2 のビット 1 2 5 : 6 3 が、マルチプレクサ 1 6 0 2 を介して、選択された疎サブツリー記述子 1 6 1 0 上で変換論理 1 6 0 4 へ転送される。

【 0 1 4 0 】

【表 3】

疎サブツリー記述子セレクト	サブツリーデータ	疎サブツリー記述子
0000	ビット 62:0	1
0001	ビット 125:63	2
0010	ビット 188:126	3
0011	ビット 251:189	4

表 3

【 0 1 4 1 】

サブツリーメモリ 4 0 0 内の各サブツリーエントリ 4 0 4 はスパースモードまたはデンスモードで構成され得る。スパースモードで構成された各サブツリーエントリ 4 0 4 は、モード 1 4 0 4 を介して疎サブツリー記述子 1 4 0 0 ごとに異なる数のノード記述子 1 4 0 2 を記憶するよう構成され得る。スパースモードで構成されたサブツリーエントリ 4 0 4 内の疎サブツリー記述子 1 4 0 0 はすべて、疎サブツリー記述子 1 4 0 0 ごとに同数のノード記述子 1 4 0 2 を記憶する。

【 0 1 4 2 】

ノード記述子 1 4 0 2 はサブツリーにおける複数のノードを表わすようにコード化され得る。ノード記述子 1 4 0 2 によって表わされる複数の 8 ビットノードは、8 ビットのうちのいくつかをマスキングすることにより識別される。マスクビットを各ノード記述子 1 4 0 2 で記憶する代わりに、9 個のビットノード記述子 1 4 0 2 を用いてノード記述子 1 4 0 2 が表わす 8 ビット幅のノードを完全に符号化する。この 8 ビット幅のノードは、ランビットレングス符号化を用いて 9 ビット幅のノード記述子 1 4 0 2 において符号化される

。ランビットレングス符号化により、そのノードの8ビットのうちのいずれがマスキングされるかが識別できるようになる。

【0143】

変換論理1604は、選択された疎サブツリー記述子1400に記憶された9ビット幅のノード記述子1402¹ - 1402ⁿを、「X」（ドントケア）にセットされたビットを含む8ビットのCAM値1612へ変換し、この8ビットのCAM値1612をCAM1606へロードする。9ビットのノード記述子1402を変換論理1604により8ビットのCAM値1612へ変換する例を以下の表4に示す。

【0144】

【表4】

8ビット値	9ビットコード
101100XX	101100100
100XXXXX	100100000
001111XX	001111100
0011XXXX	001110000

表4

【0145】

9ビットコードの列は、ノード記述子1402に記憶された値を表わす。表4の第1行目を見ると、ノード記述子1402に記憶された9ビットコードは「101100100」であり、対応する8ビット値「101100XX」がCAM1606に記憶される。変換論理1604は、9ビットのコードを「1」にセットされた第1のビットに対して右から左へサーチすることにより、この9ビットコードを変換する。9ビットコード中のビットを右から左へ見ると、最初の2つのビットが「0」にセットされ、3番目のビットが「1」にセットされている。変換論理1604は、最初の「1」の右側に「0」が2つあるので、「100」を2つのドントケア（「XX」）に変換する。1つ目の「1」は無視され、残りのビットが8ビット値の次の各ビットへと直接コピーされる。

【0146】

表4の第2行目を見ると、ノード記述子1402に記憶された9ビットコードは「100100000」である。変換論理1604は、最初の「1」についてこの9ビットコードを右から左へサーチすることにより、この9ビットコードを変換する。5番目の数字が「1」を記憶している。この9ビットコードは、5つの最下位ビット（「LSB」）が「ドントケア」（「X」）にセットされた8ビット値に変換される。9ビットのランビットレングス符号化を用いてノード記述子1402を記憶することにより、各ノード記述子1402ごとに要求されるビット数は最小になり、よって、ルックアップテーブル200中に記憶され得るノード記述子1402の数が増加する。

【0147】

9ビットのノード記述子1402を8ビット値に変換した後、変換論理1604はこの8ビット値をCAM1606内へロードする。この8ビット値は、ノード記述子1402が選択された疎サブツリー記述子1400に記憶されたのと同じ順序で、すなわち最短一致から最長一致の順序で、CAM1606内にロードされる。CAM1606は、疎サブツリー記述子1400ごとのノード記述子1402の最大数を記憶するための記憶部を提供する。したがって、CAM1606は8ビット幅×16エントリの深さであり、デフォルトマップアドレスおよびモード5疎サブツリー記述子1400に対する最大数のノード記述子1402を記憶するために15のエントリを提供する。CAM1606にはターナリ能力（ternary capability）と複数一致リゾルバ（multi-match resolver）とが内蔵されている。真の連想メモリを提供するのとは対照的に、小さなサイズのCAM1606が

10

20

30

40

50

、ゲートの形態で実現され得る。すなわち、CAM 1606は、CAMをエミュレートするハードウェア回路で実現され得る。

【0148】

疎サブツリー記述子1400に記憶されたノード記述子1402の数は、疎サブツリー記述子1400が記憶されるサブツリーエントリ404を決定する。特定のモードの範囲内のノード記述子1402を記憶する疎サブツリー記述子1400は、同じサブツリーエントリ404に記憶される。各サブツリーのデフォルトルートに対して、デフォルトマップアドレスが算出される。デフォルトの8ビット値は、CAM 1606内の第1の場所に永久に記憶され、デフォルトマップアドレスを算出する。

【0149】

選択された疎サブツリー1400に対する8ビット値がCAM 1606内へロードされた後、CAM 1606はキー210bの次の部分でサーチされる。キー210bの次の部分において最大数のビットに一致するCAM 1606内のエントリが選択される。CAMのサーチの結果得られた一致アドレスは、ブロックオフセット714として転送される。ブロックオフセット714を用いて、サブツリーマップ418(図5)に記憶されたルートに対応するマップエントリのマップアドレス416が決定される。

【0150】

図19A-Dは、疎に配置されたサブツリー1700内のあるノードに対するブロックオフセット714の選択を図示する。図17Aは疎に配置されたサブツリー1700におけるルートを図示したものである。サブツリー1700内のノードは3つのルートr0、r1およびr2のうちの1つに対応しており、r0はサブツリー1700のデフォルトルートである。2つのルートr1およびr2は、疎サブツリー記述子1400のノード記述子1402¹および1402²で符号化される。デフォルトルートr0の値は、CAM 1606内の第1のエントリ1702に永久に記憶される。表2を参照して、2つのノード記述子1402を備える疎サブツリー記述子1400は、サブツリーエントリ404に記憶され、モードフィールド1404は「1」にセットされる。

【0151】

サブツリー1700を見ると、r2は10XXXXXXに一致するすべてのノードに対応し、r1は010XXXXXに一致するすべてのノードに対応する。各ノード記述子1402¹および1402²により要求されるビット数を最小にして疎サブツリー記述子1400中の各ルートを記述するために、ノード記述子1402¹および1402²は、ランビットレングス符号化を用いてコード化される。コード化の方法は、そのノードを完全に符号化するのに用いられるビット数より1ビット多いものを用いる。最初の「X」(「ドントケア」)の場所に「1」が挿入され、残りのXは0としてコード化される。すなわち、ルート10XXXXXXは10100000に翻訳され、010XXXXXは010100000に翻訳される。

【0152】

図19Bは、疎サブツリー記述子1400中のノード記述子1402¹および1402²の記憶部を表わす。ノード記述子1402¹および1402²はサブツリーエントリ404に記憶され、モードフィールド1404は「1」にセットされる。これは、疎サブツリー記述子1400に記憶された2つのノード記述子1402¹および1402²があるからである。サブツリーの最長一致はr1である。なぜなら、r1は最初の3ビットの一致を要求し、r2は最初の2ビットの一致を要求するからである。ノード記述子1402¹および1402²は疎サブツリー記述子1400において最短一致から最長一致の順序で記憶され、最初にr2のノード記述子1402¹が記憶されて次にr1のノード記述子1402²が次に記憶される。

【0153】

図19Cは、ノード記述子1402²の8ビットのマスキングされた値1706への変換を表わす。ノード記述子ビット1708¹-1708⁹を左から右へ見ると、最初の「1」がビット1708⁶に記憶され、これは8ビットのマスキング値1706のマスキングビット

10

20

30

40

50

の終わりを記している。ノード記述子 1402^2 を 8 ビットのマスキングされた値 1706 に変換するために、以下のビット変換が行なわれる。ノード記述子のビット 1708^1 に記憶された「0」は「X」に変換され、8 ビットのマスキングされた値のビット 1710^1 に記憶される。ノード記述子のビット 1708^2 に記憶された「0」は「X」に変換され、8 ビットのマスキングされた値のビット 1710^2 に記憶される。ノード記述子のビット 1708^3 に記憶された「0」は「X」に変換され、8 ビットのマスキングされた値のビット 1710^3 に記憶される。ノード記述子のビット 1708^4 に記憶された「0」は「X」に変換され、8 ビットマスキングされた値のビット 1710^4 に記憶される。ノード記述子のビット 1708^5 に記憶された「0」は「X」に変換され、8 ビットのマスキングされた値のビット 1710^5 に記憶される。ノード記述子のビット 1708^6 に記憶された「1」は無視される。ノード記述子のビット 1708^7 に記憶された「0」は 8 ビットのマスキングされた値のビット 1710^6 に記憶される。ノード記述子のビット 1708^8 に記憶された「1」は 8 ビットのマスキングされた値のビット 1710^7 に記憶される。ノード記述子のビット 1708^9 に記憶された「0」は 8 ビットのマスキングされた値のビット 1710^8 に記憶される。

【0154】

図 19D は、CAM 1606 内のノード記述子 1402^1 および 1402^2 の記憶部と、選択された疎サブツリー記述子 1400 に対してサブツリーマップ 418 (図 5) に記憶された対応のマッパエントリ $504^1 - 504^3$ とを表わす。選択されたサブツリー記述子 1400 に記憶された 9 ビットのノード記述子 1402^1 および 1402^2 は変換論理 1604 (図 18) に変換され、CAM 1606 内へロードされる。CAM 1606 内の第 1 のエントリ 1702 は、図 19A のサブツリー 1700 に示す $r0$ に対するデフォルトエントリである。第 2 のエントリ 1704 は、選択された疎サブツリー記述子 1400 に記憶された第 1 のノード記述子 1402^1 から変換される。第 2 のエントリ 1704 は、 $r2$ に対して変換される最短の一致である。選択されたサブツリー記述子 1400 に記憶された第 2 のノード記述子 1402^2 は、 010100000 から $010XXXXXX$ へ変換され、CAM 1606 内の第 3 のエントリ 1706 に記憶される。

【0155】

CAM 1606 のサーチの結果、ブロックオフセット 714 (図 18) が得られる。このブロックオフセット 714 は、サブツリーマップ 418 (図 5) に記憶されたマッパエントリ $504^1 - 504^3$ のマッパアドレス 416 を決定するのに用いられる。CAM 1606 は、最長一致を記憶するエントリ 1702 、 1704 および 1706 についてキー $210b$ の第 2 の部分でサーチされる。CAM 1606 により提供されたブロックオフセット 714 は、選択されたサブツリーエントリ 404 内のポインタフィールド 408 に記憶されたブロックベースアドレス 600^1 、 600^1 のうちの 1 つに依存するサブツリーベースアドレスと組合せられる。

【0156】

図 20 は、図 8 に示すポインタ論理 702 中のスパースモードベースセレクト論理 1800 を表わすブロック図である。ポインタロジック 702 は、サブツリーマップ 418 (図 5) 内のマッパエントリ 504 (図 6B) のマッパアドレス 416 を算出するのに用いられるベースアドレス 716 を選択する。ポインタ論理 702 はデンスモードベースセレクト論理 710 およびスパースモードベースセレクト論理 1800 を含み、それらのうちの 1 つが、前のマッパレベルから転送された、サブツリーエントリ記述子 304 (図 4) に記憶されたタイプ 1406 の状態に依存して、選択される。既に述べたように、タイプ 1406 の状態は、サブツリーエントリ 404 がデンスモードで構成されるか否かを示す。

【0157】

スパースモードベースセレクト論理 1800 は、サブツリーエントリ 404 が複数の疎サブツリー記述子 1400 を記憶していれば、疎サブツリー記述子 1400 のベースアドレス 716 を算出する。スパースモードベースセレクト論理 1800 は、モードフィールド 1404 に記憶されたモード値 1608 と、サブツリーエントリ 404 内のブロックベー

10

20

30

40

50

スアドレスフィールド 600^1 および 600^2 に記憶されたサブツリーポインタ 414 と、前のマッパレベルから転送されたサブツリーエントリ記述子 304 (図4) に記憶された疎サブツリー記述子セレクト 1408 とを用いて、ベースアドレス 716 を算出する。ベースアドレス 716 は以下のように算出される。

【0158】

(疎サブツリー記述子の) ベースアドレス = ブロックベースアドレス + ベースオフセット
ここでベースオフセット = $((1 + \text{ノード / サブツリー}) * \text{疎サブツリー記述子セレクト})$

たとえば、スパースモード4で構成されたサブツリーエントリ 404 におけるサブツリー番号2の開始時のベースアドレス 716 を見つけるためには、最初にベースオフセットが計算される。サブツリー番号2に対する疎サブツリー記述子セレクト 1408 は「1」であり、ノード / サブツリーの数 7 である(表2参照)。ベースオフセットは $8((1 + 7) * 1)$ である。ブロックベースアドレス 600^1 および 600^2 の各々が、サブツリーエントリ 404 に対して割当てられた 16 のマッパアドレスのブロックのベースアドレスである。サブツリー番号2のベースオフセットは 8 であり、これは 16 より小さいので、サブツリー2のブロックベースアドレスはブロックベースアドレス 600^1 になり、疎サブツリー記述子のベースアドレス 716 はブロックベースアドレス $600^1 + 8$ になる。以下の表5に、モード4で構成されたサブツリーエントリ 404 における4つのサブツリーの各々についてのサブツリーベースアドレスを表わす。

【0159】

【表5】

サブツリーベースアドレス	サブツリー
ブロックベースアドレス $1+0$	1
ブロックベースアドレス $2+8$	2
ブロックベースアドレス $2+0$	3
ブロックベースアドレス $2+8$	4

表5

【0160】

図21は、サブツリーメモリ 400 に記憶された稠密サブツリー記述子および疎サブツリー記述子を表わす。図21は図15と関連して説明する。サブツリーB(図21)の稠密サブツリー記述子がサブツリーエントリ 404^1 内のデータフィールド 406^1 に記憶される。サブツリーA(図21)の疎サブツリー記述子 1400^1 は、サブツリーエントリ 404^2 内のデータフィールド 406^2 に記憶される。稠密サブツリー記述子は、図6Bに関連して既に述べたように、サブツリーBの最下レベルにおいて各ノードに対してノードビットを記憶する。スパースモード記述子 1400^1 は、図19Bに関連して既に述べたように、ルート $r4$ および $r5$ に対応するノード記述子 1402^1 および 1402^2 を含む。サブツリーインデックス 312 はサブツリーエントリ 404^1 および 404^2 を選択する。

【0161】

$s0$ (図15) に対してマッパ $106a$ 内のマッパエントリ 504 中のサブツリーエントリ記述子 304 (図4) に記憶されたサブツリーインデックス 312 がサブツリーエントリ 404^1 を選択する。 $s1$ (図15) に対するマッパ $106a$ 内のマッパエントリ 504 (図6B) におけるサブツリーエントリ記述子 304 (図4) に記憶されたサブツリーインデックス 312 は、サブツリーエントリ 404^2 を選択する。このように、サブツリーメモリ 400 は、疎サブツリーおよび稠密サブツリーに対してサブツリーエントリ 404^1 および 404^2 を記憶し得る。

【 0 1 6 2 】

図 2 2 は、疎に配置されたサブツリーおよび密に配置されたサブツリーにおけるあるノードに対するルートを記憶するサブツリーマップ 4 1 8 (図 5) 内のマップエントリ 5 0 4 (図 6 B) についてのマップアドレス 4 1 6 (図 5) を提供するための方法を図示するフローチャートである。いずれのサブツリーエントリ 4 0 4 も、複数の疎サブツリー記述子または単一の稠密サブツリー記述子を記憶し得る。バイナリツリー内でルートがいかに分散されるかに依存して、疎サブツリー記述子と密サブツリー記述子とのいかなる組合せも可能である。サブツリーメモリ 4 0 0 内のサブツリーエントリ 4 0 4 でスパースモードと稠密サブツリー記述子とを混合させ一致させる柔軟性により、サブツリーメモリ 4 0 0 をより有用化することが可能になる。

10

【 0 1 6 3 】

ステップ 1 9 0 0 において、選択されたサブツリーエントリ 4 0 4 の構成は、前のマップレベルで選択されたサブツリーエントリ記述子 3 0 4 (図 4) に記憶されたタイプ 1 4 0 6 (図 1 6 C) の状態から決定される。サブツリーエントリ 4 0 4 タイプがスパースモードで構成される場合、処理はステップ 1 9 0 2 へ進む。そうでなければ、処理はステップ 1 9 1 4 へ進む。

【 0 1 6 4 】

ステップ 1 9 0 2 において、サブツリーエントリ 4 0 4 はスパースモードで構成される。スパースモードで構成されるサブツリーエントリ 4 0 4 は、複数の疎サブツリー記述子 1 4 0 0 を記憶する。サブツリーエントリ 4 0 4 に記憶された疎サブツリー記述子 1 4 0 0 の数は、モードフィールド 1 4 0 4 の状態に依存する。オフセット論理 7 0 0 におけるスパースモード論理 1 5 0 2 により、図 1 4 に関連して前に述べたように前のマップレベルから転送されたサブツリーエントリ記述子 3 0 4 (図 4) に記憶された疎サブツリー記述子セレクト 1 4 0 8 とモードフィールド 1 4 0 4 の内容とに依存して、サブツリーエントリ 4 0 4 から疎サブツリー記述子 1 4 0 0 が選択される。処理はステップ 1 9 0 4 へ進む。

20

【 0 1 6 5 】

ステップ 1 9 0 4 において、選択された疎サブツリー記述子 1 4 0 0 内のノード記述子 1 4 0 2 に記憶された 9 ビットコード化された値は 8 ビット値に変換され、最短一致から最長一致の順序で、CAM 1 6 0 6 に記憶される。処理はステップ 1 9 0 6 へ進む。

30

【 0 1 6 6 】

ステップ 1 9 0 6 において、CAM 1 6 0 6 が、最長の一致を記憶する CAM エントリについてキー 2 1 0 b の次の部分でサーチされる。処理はステップ 1 9 0 8 へ進む。

【 0 1 6 7 】

ステップ 1 9 0 8 において、キー 2 1 0 b の次の部分の最長一致を記憶する CAM 1 6 0 6 における場所のアドレスが、ブロックオフセット 7 1 4 として転送される。ブロックオフセット 7 1 4 を用いて、サブツリーマップ 4 1 8 (図 5) 内のマップエントリ 5 0 4 (図 6 B) のマップアドレス 4 1 6 (図 5) が算出される。処理はステップ 1 9 1 0 へ進む。

【 0 1 6 8 】

ステップ 1 9 1 0 において、選択された疎サブツリー記述子 1 4 0 0 のベースアドレス 7 1 6 (図 2 0) は、前のマップレベルから転送されたサブツリーエントリ記述子 3 0 4 (図 4) に記憶された疎サブツリー記述子セレクト 1 4 0 8 と、選択されたサブツリーエントリ 4 0 4 に記憶されたモードフィールド 1 4 0 4 の内容とに依存して、算出される。処理はステップ 1 9 1 2 へ進む。

40

【 0 1 6 9 】

ステップ 1 9 1 2 において、マップアドレス 4 1 6 は、アダー論理 7 0 4 (図 8) 内でブロックオフセット 7 1 4 とベースアドレス 7 1 6 とを加算することにより算出される。サブツリーマップ 4 1 8 (図 5) 内のマップアドレス 4 1 6 により識別されたマップエントリ 5 0 4 (図 6 B) は、ルートエントリ 3 0 2 (図 4) またはサブツリーエントリ記述子

50

304 (図4)のいずれかを記憶する。マップエントリ504 (図6B)がルートエントリ302 (図4)を記憶する場合、サーチは完了する。マップエントリ504 (図6B)がサブツリーエントリ記述子304 (図4)を記憶する場合、キー210に対応する値のサーチが次のマップレベルにおいて継続される。

【0170】

ステップ1914において、サブツリーエントリ404はデンスモードで構成され、データフィールド406に単一の稠密サブツリー記述子を記憶する。ブロックオフセット714は、図6Bに関連して前述したように、サブツリーエントリ404内のデータフィールド406に記憶された稠密サブツリー記述子に記憶された「1」の数をカウントすることにより、算出される。処理はステップ1916へ進む。

10

【0171】

ステップ1916において、サブツリーエントリ404は、サブツリーエントリ404内のポインタフィールド408に16個のブロックベースアドレス600を記憶する。ブロックベースポインタ600のうちの1つが、図8に関連して前述したポインタ論理702におけるデンスモードベースセレクト論理710により選択される。処理はステップ1912へ進む。

インクリメンタルな更新

図23は、ルックアップテーブル200に追加される新たなルートのバイナリツリーを表わす図である。このバイナリツリーは、マップレベル__1 2000、マップレベル__2 2002およびマップレベル__3 2004について、ルックアップテーブル200に記憶されたルートを表わす。マップレベル__2 2002は、サブツリーAおよびBに対するルートを記憶する。マップレベル__3 2004はサブツリーA₁、A₂、B₁およびB₂に対するルートを記憶する。s5は、サブツリーマップ418 (図5)に記憶されたサブツリーエントリ記述子304 (図4)を表わす。s5に対するサブツリーエントリ記述子304 (図4)はサブツリーB₂に対するポインタを記憶し、キー210に対する最長一致のルートのサーチをマップレベル__3 2004で継続できるようにする。

20

【0172】

サブツリーB₂ 2006は、2つのルート、r6およびh1のみを有するので、疎サブツリーである。したがって、ノードr6およびh1のノード記述子1402 (図16A)は、図14Aに関連して既に述べたように、疎サブツリー記述子1400に記憶される。サブツリーB₂ 2006の疎サブツリー記述子1400はサブツリーエントリ404に記憶され、サブツリーメモリ400においてモードフィールド1404は1にセットされる。これは、疎サブツリー記述子1400に記憶されたノード記述子1402が2つあるためである。

30

【0173】

サブツリーB₂ 2008に示す新たなルートh2は、ルックアップテーブル200に追加されることになる。新たなルートh1をルックアップテーブル内のサブツリーB₂ 2006に直接追加することはできない。なぜなら、サブツリーB₂ 2006に対するルートを追加することにより、疎サブツリー記述子1400に記憶されたノード記述子1402の数が2から3へ増加するからである。疎サブツリー記述子1400へノード記述子1402を追加するとモードフィールド1404が「2」にセットされたサブツリーエントリ404において新たな疎サブツリー記述子1400を割当てて必要が生じる。すなわち、新たなルートh1の追加により、サブツリーB₂ 2006をサブツリーB₂ 2008で置き換える必要が生じる。

40

【0174】

図24は、プロセッサメモリ2400に記憶された更新ルートを表わす。ルックアップテーブル200に記憶されたバイナリツリーのコピーもまた、ルックアップテーブル200とは別のプロセッサメモリ2400に記憶される。サブツリーB₂ 2006について記憶されたルートはプロセッサメモリ2400内のサブツリーB₂ 2008にコピーされ、新たなルートh2がサブツリーB₂ 2008に追加される。

50

【 0 1 7 5 】

ルート更新ルーチン 2 4 0 2 は、ルート更新命令 2 4 0 4 のシーケンスを生成してサブツリー B_2 2 0 0 8 をルックアップテーブル 2 0 0 に追加し、ルート更新 2 4 0 4 をテーブル更新ルーチン 2 4 0 6 へ転送する。テーブル更新ルーチン 2 4 0 6 はルート更新 2 4 0 2 に対してテーブル更新 2 4 1 0 を生成し、更新サイクル 2 4 1 2 を転送し、ルックアップテーブル 2 0 0 をルート更新 2 4 0 4 で更新する。更新サイクル 2 4 1 2 は、サブツリーマップ 4 1 8 (図 5) およびサブツリーメモリ 4 0 0 (図 5) における適切なメモリ場所にルート更新を書込む。

【 0 1 7 6 】

図 2 3 を参照して、更新サイクル 2 4 1 2 はサブツリーマップ 4 1 8 (図 5) の一部分を割当て、新たなサブツリー B_2 2 0 0 8 のルートをマップエントリ 5 0 4 (図 6 B) に記憶するための命令を含む。サブツリー B_2 2 0 0 8 は、ルート h_1 および r_6 ならびに新たなルート h_2 に対してマップエントリ 5 0 4 (図 6 B) に記憶されたルートエントリを含む。サブツリー B_2 2 0 0 8 に対するルートエントリがサブツリーマップ 4 1 8 (図 5) 内のマップエントリ 5 0 4 (図 6 B) に記憶された後、ルートに対するノード記述子 1 4 0 2 が作成され、疎サブツリー記述子 1 4 0 0 に記憶される。疎サブツリー記述子 1 4 0 0 はサブツリーエントリ 4 0 4 に記憶される。サブツリーエントリ 4 0 4 のモード 1 4 0 4 は、疎サブツリー記述子 1 4 0 0 に記憶されたノード記述子 1 4 0 2 の数に関連する。

【 0 1 7 7 】

サブツリー B_2 2 0 0 8 の疎サブツリー記述子 1 4 0 0 がルックアップテーブル 2 0 0 中のサブツリーメモリ 4 0 0 のサブツリーエントリ 4 0 4 に記憶された後、 s_5 で表わされるサブツリーエントリ記述子 3 0 4 (図 4) は、サブツリー B_2 2 0 0 6 の代わりにサブツリー B_2 2 0 0 8 を指すように変更される。サブツリー B_2 2 0 0 8 がルックアップテーブルに追加される一方、サブツリー B_2 2 0 0 6 に記憶されたルート r_6 および h_1 は s_5 を介してアクセスされ得る。サブツリー B_2 2 0 0 8 がルックアップテーブルに記憶された後、 s_5 は、サブツリー B_2 2 0 0 8 およびルート r_6 を指すように変更され、新たなルート h_2 がアクセスされ得る。したがって、サブツリー B_2 2 0 0 6 の、ルート r_6 および h_1 に対応するルートインデックスについてのサーチを継続することができ、一方、新たなルート h_2 はルックアップテーブル 2 0 0 に追加される。

【 0 1 7 8 】

図 2 5 は、ルックアップテーブル 2 0 0 中のサブツリーマップ 4 1 8 b におけるマップエントリ 5 0 4 c^4 に記憶された、図 2 3 に示す新たなルート h_2 を表わす。図 2 5 は、図 2 4 に示すバイナリツリーの図に関連して説明する。

【 0 1 7 9 】

マップレベル $_2$ 2 0 0 2 におけるサブツリー B は 3 つのルート、すなわち r_3 、 s_4 および s_5 を有する。サブツリー B は 1 6 未満のルートを有するので、疎サブツリーである。サブツリー B の r_3 、 s_4 および s_5 に対するノード記述子 1 4 0 2 a^1 - 1 4 0 2 a^3 は、サブツリーメモリ 4 0 0 a 内のサブツリーエントリ 4 0 4 a における疎サブツリー記述子 1 4 0 0 a^1 に記憶される。マップエントリ 5 0 4 a^2 - 5 0 4 a^4 は、サブツリー B 内の各ルートに対してサブツリーマップ 4 1 8 a に記憶される。サブツリー B のデフォルトルートは、サブツリーマップ 4 1 8 a 内のマップエントリ 5 0 4 a^1 に記憶される。マップエントリ 5 0 4 a^2 - 5 0 4 a^4 の各々は、そのノードに対するルートエントリ 3 0 2 (図 4) またはサブツリーエントリ記述子 3 0 4 (図 4) を記憶する。サブツリーエントリ記述子 3 0 4 (図 4) は、ルート s_4 については 5 0 4 a^3 に、 s_5 については 5 0 4 a^4 に記憶される。 s_5 についてマップエントリ 5 0 4 a^4 に記憶されたサブツリーエントリ記述子 3 0 4 (図 4) は、サブツリーメモリ 4 0 0 b にサブツリーインデックス 3 1 2 b を与え、次のレベルのサーチ、すなわちマップレベル $_3$ 2 0 0 4 に対するサーチを開始する。

【 0 1 8 0 】

10

20

30

40

50

サブツリー B_2 は、2つのルートすなわち h_1 および r_6 を有するので、これもまた疎サブツリーである。ノード記述子 $1402b^1 - 1402b^2$ は、サブツリーメモリ $400b$ 内のサブツリーエントリ $404b$ における疎サブツリー記述子 $1400b^1$ に記憶される。サブツリー B_2 内の各ルートは、マップエントリ $504b^2 - 504b^3$ に記憶され、サブツリー B_2 のデフォルトルートがマップエントリ $504b^1$ に記憶される。

【0181】

サブツリー B_2 2006 内でルート h_1 についてサーチするために、ルート s_5 についてノード記述子 1402 を記憶する疎サブツリー記述子 $1400a$ を記憶しているサブツリーエントリ $404a$ のアドレスが、サブツリーインデックス $312a$ 上でサブツリーメモリ $400a$ へ転送される。選択されたサブツリーエントリ $404a$ に記憶されたデータフィールド 406 およびポインタフィールド 408 は、サブツリーデータ $412a$ およびサブツリーポインタ $414a$ 上でマップアドレス論理 $402a$ へ転送される。マップアドレス論理 $402a$ は、 s_5 についてサブツリーエントリを記憶しているマップエントリ $504a^4$ のマップアドレス $416a$ を生成する。マップアドレス $416a$ は、サブツリーデータ $412a$ 、サブツリーポインタ $414a$ およびキー $210b$ の次の部分に依存する。 s_5 のサブツリーエントリは、サブツリーインデックス $312b$ 上でサブツリーメモリ $400b$ へ転送される。

10

【0182】

サブツリーメモリ $400b$ は、サブツリー B_2 2006 についてノード記述子 $1402b^2$ および $1402b^1$ を記憶する。 B_2 についての疎サブツリー記述子 $1400b^1$ がサブツリーエントリ $404b$ に記憶される。サブツリーエントリ $404b$ に記憶されたデータフィールド 406 およびポインタフィールド 408 は、サブツリーデータ $412b$ およびサブツリーポインタ $414b$ 上でマップアドレス論理 $402b$ へ転送される。マップアドレス論理 $402b$ は、 h_1 についてルートエントリを記憶しているマップエントリ $504b^3$ のマップアドレス $416b$ を生成する。マップアドレス $416b$ は、サブツリーデータ $412b$ 、サブツリーポインタ $414b$ およびキー $210c$ の次の部分に依存している。

20

【0183】

ルート h_2 をサブツリー B_2 2006 に追加するためには、サブツリーマップ $418b$ 内のこれまでに未使用のマップエントリ $602b$ のブロックが割当てられ、サブツリー B_2 2008 についてルート r_6 、 h_1 および h_2 を記憶しているマップエントリ $504c^2 - 504c^4$ を記憶する。マップエントリ $504c^1$ は、サブツリー B_2 2008 のデフォルトエントリ、すなわち、マップエントリ $504b^1$ に記憶されたものと同じ値を記憶する。マップエントリ $504c^2$ はルート r_6 のルートエントリ、すなわちマップエントリ $504b^2$ に記憶されたものと同じ値を記憶する。マップエントリ $504c^3$ はルート h_1 のルートエントリ、すなわちマップエントリ $504b^3$ に記憶されたものと同じ値を記憶する。マップエントリ $504c^4$ は、新たなルート h_2 に対するルートエントリを記憶する。マップエントリ $504c^{1-4}$ のブロックが書込まれている間に、マップエントリ $504b^1 - 504b^3$ に記憶されたルートエントリは、サブツリーマップ $418a$ 内の $504a^4$ にルート s_5 について記憶されたサブツリーエントリを介して、アクセスされ得る。

30

40

【0184】

サブツリーマップ $418b$ にはサブツリー B_2 2008 についてのマップエントリ $504c^{1-4}$ が記憶されており、疎サブツリー記述子 $1400c^1$ がサブツリーメモリ $400b$ に追加される。ノード記述子 $1402c^{1-3}$ の数は16未満であるので、ノード記述子 $1402c^{1-3}$ は疎サブツリー記述子 $1400c^1$ に記憶される。サブツリーメモリ $400b$ 内のサブツリー記述子 1400^1 の場所は、疎サブツリー記述子 $1400c^1$ に関連付けられるノード記述子 $1402c^{1-3}$ の数に依存する。サブツリー B_2 2006 に新たなルートを追加することにより、疎サブツリー記述子 $1400c^1$ に対して記憶されるノード記述子 $1402c^1 - 1402c^3$ の数は2から3へ増加する。疎サブツリー記述子 1400

50

c^1 はサブツリーエントリ404cに記憶され、各疎サブツリー記述子ごとに3つのノード記述子があり、モードフィールド1404は「2」にセットされる。疎サブツリー記述子1400 c^1 は、利用可能なスペースがあれば現在のモード3サブツリーエントリ404cに記憶されるか、または新たなモード3サブツリーエントリが割当てられる。 B_2 2008内のルートについてのノード記述子は、モード3サブツリーエントリ404cの疎サブツリー記述子1400 c^1 におけるノード記述子1402 c^{1-3} に記憶される。

【0185】

疎サブツリー記述子1400 c^1 およびノード記述子1402 c^{1-3} がサブツリーメモリ400bに記憶された後、サブツリー B_2 2008がアクセスされ得る。 B_2 2008にアクセスを提供するために、サブツリーエントリ504 a^4 は、サブツリーエントリ404b内の疎サブツリー記述子1400 b^1 の代わりにサブツリーエントリ404c内の疎サブツリー記述子1400 c^1 をインデックス付けするよう変更される。マップエントリ504 c^4 に記憶されたルートh2についてのルートエントリならびにそれぞれのマップエントリ504 c^2 および504 c^3 に記憶されたルートr6およびh1がアクセスされ得る。

【0186】

マップエントリ504 b^1 -504 b^3 はもはやアクセス不可能であるので、割当解除され、将来の割当のためにフリーリスト(図示せず)上に置かれる。また疎サブツリー記述子1400 b^1 ももはやアクセス不可能である。こうして、疎サブツリー記述子1400 b^1 は割当解除され、将来の割当のためにフリーリスト(図示せず)に置かれる。

【0187】

疎サブツリーにルートを追加することについて説明した。ルートは、新たに割当てられたサブツリーエントリ404に新たな稠密サブツリー記述子を記憶し、対応のマップエントリをサブツリーマップ418に記憶することにより、また、マップエントリ504 a^4 に記憶されたサブツリーエントリを新たに割当てられたサブツリーエントリ404のインデックス付けを行なうように変更することにより、追加することもできる。

【0188】

図26は、図25に示すルックアップテーブル200にあるルートを追加するためにインクリメンタルな更新を行なうためのステップを表わすフローチャートである。

【0189】

ステップ2200において、サブツリーごとのルート数が算出され、ルート更新結果が疎サブツリーであるか稠密サブツリーであるかが判定される。ルート更新後のサブツリーが稠密であれば、処理はステップ2218へ進む。ルート更新後のサブツリーが疎であれば、処理はステップ2202へ進む。

【0190】

ステップ2202において、サブツリーは疎である。疎サブツリーモードが決定される。処理は2204へ進む。

【0191】

ステップ2204において、サブツリーマップ418(図5)に記憶された部分的にフルにされたサブツリーエントリ404のリストがサーチされ、新たな疎サブツリー記述子1400 c^1 が前に割当てられたサブツリーエントリ404に記憶され得るか否かが判定される。たとえば、4つの疎サブツリー記述子1400 c^1 -1400 c^4 がモード4サブツリーエントリ404に記憶され得る。3つのみが記憶される場合、サブツリーエントリ404は部分的にフルにされており、部分的に満たされたサブツリーエントリ404のリストに記憶される。部分的にフルなサブツリーエントリ404が利用可能であれば、処理はステップ2208へ進む。そうでなければ、処理はステップ2206へ進む。

【0192】

ステップ2206において、新たなサブツリーエントリ404cが疎サブツリー記述子1400 c^1 を記憶するために割当てられ、マップエントリ504 c^1 -504 c^4 が、新たに割当てられたサブツリーエントリ404c内の疎サブツリー記述子1400 c^1 に記憶

10

20

30

40

50

されたノード記述子 $1402c^1 - 1402c^3$ のマップエントリ 504 (図6B) を記憶するために、サブツリーマップに割当てられる。サブツリーマップ 418 (図5) 内のマップエントリ $504c^1 - 504c^4$ の割当てられたブロックに対するポインタは、新たなサブツリーエントリ $404c$ 内のポインタフィールド 408 に記憶される。処理はステップ 2208 へ進む。

【0193】

ステップ 2208 において、疎サブツリー記述子 $1400c^1$ に対するサブツリーマップ内の第1のマップエントリ $504c^1$ の場所が、サブツリーエントリ $404c$ 内のポインタフィールド 408 に記憶されたポインタおよびサブツリーエントリ $404c$ 内のモードフィールド 1404 に記憶されたモードから判定される。処理はステップ 2210 へ進む。

10

【0194】

ステップ 2210 において、疎サブツリーのルートエントリが、サブツリーマップ $418b$ 内のマップエントリ $504c^1 - 504c^4$ に記憶される。処理はステップ 2212 へ進む。

【0195】

ステップ 2212 において、ノード記述子 $1402c^1 - 1402c^3$ を記憶している疎サブツリー記述子 $1400c^1$ がサブツリーエントリ $404c$ に記憶される。処理はステップ 2214 へ進む。

【0196】

20

ステップ 2214 において、マップエントリ $504a^4$ に記憶されたサブツリーエントリ記述子 304 (図4) は、サブツリーエントリ $404c$ に記憶された新たな疎サブツリー記述子 $1400c^1$ のインデックス付けを行なうよう変更される。マップエントリ $504c^4$ に記憶された $h2$ のルートエントリがここでアクセス可能にある。処理はステップ 2216 へ進む。

【0197】

ステップ 2216 において、マップエントリ $504b^1 - 504b^3$ および疎サブツリー記述子 $1400b$ はもはやアクセス不可能である。マップエントリ $504b^1 - 504b^3$ は、サブツリーマップ $418b$ のマップエントリ 504 (図6B) のフリーリストに置かれ、他のルートを記憶するために割当てられ得る。サブツリーエントリ $404b$ 内の第1の利用可能な場所は、部分的にフルなサブツリーエントリのリストにおいて更新される。処理が完了する。

30

【0198】

ステップ 2218 において、新たなサブツリーエントリ 404 がプロセッサメモリ 2400 (図24) に記憶されたフリーのサブツリーエントリ 404 のリストから割当てられる。この新たなサブツリーエントリ 404 は、新たな稠密サブツリー記述子を記憶するために割当てられる。サブツリーマップ $418b$ におけるマップエントリ 504 (図6B) のブロックが、ルートを記憶するために割当てられる。割当てられたマップエントリ 504 (図6B) のブロックに対するポインタは、サブツリーエントリ 404 (図5) 内のポインタフィールド 408 (図7) に記憶される。処理はステップ 2220 へ進む。

40

【0199】

ステップ 2220 において、新たな稠密サブツリー記述子が、図6Aおよび6Bに関連して前述したように、新たなサブツリーエントリ 404 内のデータフィールド 406 に書込まれる。処理はステップ 2222 へ進む。

【0200】

ステップ 2222 において、稠密サブツリーのルートエントリが、サブツリーエントリ 404 内のポインタフィールド 408 に記憶されたポインタにより識別されたサブツリーマップ 418 (図5) 内のマップエントリ 504 (図6B) に記憶される。処理はステップ 2224 へ進む。

【0201】

50

ステップ 2 2 2 4 において、マップエントリ 5 0 4 a⁴に記憶されたサブツリーエントリ記述子 3 0 4 (図 4) は、新たなサブツリーエントリ 4 0 4 c に記憶された新たな稠密サブツリー記述子をインデックス付けするよう変更される。マップエントリ 5 0 4 c⁴に記憶された h₂ に対するルートエントリがここでアクセス可能になる。処理はステップ 2 2 2 6 へ進む。

【0 2 0 2】

ステップ 2 2 2 6 において、古いサブツリーエントリ 4 0 4 内のポインタフィールド 4 0 8 に記憶されたポインタによりインデックス付けされたマップエントリ 5 0 4 (図 6 B) は、プロセッサメモリ 2 4 0 0 (図 2 4) に記憶されたマップエントリのフリーリストへ戻される。この古いサブツリーエントリ 4 0 4 b は、プロセッサメモリ 2 4 0 0 (図 2 4) に記憶されたサブツリーエントリのフリーリストに追加される。

10

【0 2 0 3】

ルートをルックアップテーブルに追加する処理について述べてきた。ルートをルックアップテーブルから削除するためにも同様の処理が行なわれる。たとえば、h₂ 5 0 4 c⁴をサブツリー B₂ から削除するためには、ルート r₆ および h₁ の 2 つのノード記述子を有する新たな疎サブツリー記述子を記憶し、モード 2 サブツリーエントリに疎サブツリー記述子を記憶し、対応のサブツリーマップを更新し、マップエントリ 5 0 4 a⁴に記憶されたサブツリーエントリ記述子 3 0 4 (図 4) を、新たなサブツリーエントリ 4 0 4 に記憶された更新されたサブツリー記述子のインデックス付けを行なうよう変更する必要がある。

20

【0 2 0 4】

この発明はその好ましい実施例を参照して具体的に図示し説明してきたが、当業者には、前掲の請求項に包含される本発明の範囲から離れることなく、形式および内容の点で種々の変更をなすことができることがわかるであろう。

【図面の簡単な説明】

【図 1 A】 バイナリツリーの第 1 のレベルを表わす先行技術のビットマップを示す図である。

【図 1 B】 キャッシュメモリで実現される先行技術のルックアップテーブルを示す図である。

【図 2 A】 この発明の原則に従う最長一致プレフィックスルックアップテーブルを示す図である。

30

【図 2 B】 図 2 A に示されるルックアップテーブルに記憶されるルートインデックスのバイナリツリー表示を示す図である。

【図 3】 この発明の原則に従う 4 0 ビットキーに対する最長一致プレフィックスルックアップテーブルを示す図である。

【図 4】 図 2 A に示されるダイレクト・マップト・マップに記憶可能なマップエントリのタイプを示す図である。

【図 5】 図 2 B に示されるマップレベル 2 1 1 2 b 中のノードに対応するマップを示す図である。

【図 6 A】 サブツリーのバイナリツリー表示の図である。

40

【図 6 B】 図 6 A に示されたサブツリーのボトムレベルのノードに対応する、図 5 に示されたデータフィールドに記憶される稠密サブツリー記述子を示す図である。

【図 7】 図 5 に示された p t r フィールドを示す図である。

【図 8】 図 5 に示されたマップアドレス論理を示す図である。

【図 9】 最長一致をサーチするためのステップを図示するフローチャートの図である。

【図 1 0 A】 深度拡張を与えるための実施例を示す図である。

【図 1 0 B】 図 1 0 A に示された実施例のルックアップテーブルの 1 つを示す図である。

。

【図 1 0 C】 値を記憶するのに利用可能なマップエントリを増やすために深度拡張を与えるための別の実施例を示す図である。

50

【図 10 D】 図 10 C に示された実施例のスレーブルックアップテーブルを示す図である。

【図 11 A】 図 10 A および図 10 C に示されたルックアップテーブル中のマップエントリの中の、図 2 B に示されたルートインデックスのバイナリツリー表示のノードの分布のバイナリツリー表示を示す図である。

【図 11 B】 図 10 A および図 10 C に示されたルックアップテーブル中のマップエントリの中の、図 2 B に示されたルートインデックスのバイナリツリー表示のノードの分布のバイナリツリー表示を示す図である。

【図 12】 図 10 A および図 10 C に示されたルックアップテーブル中のマップエントリの中に値を分散するための方法を図示するフローチャートの図である。

10

【図 13】 図 10 C に示されたルックアップテーブル中のマップエントリのうち 1 つに記憶されるサーチキーに対応する値をサーチするための方法を図示するフローチャートの図である。

【図 14】 図 10 A に示されたルックアップテーブル中のマップエントリのうち 1 つに記憶されるサーチキーに対応する値をサーチするための方法を図示するフローチャートの図である。

【図 15】 第 1 のマップレベルによってインデックス付けされる第 2 のマップレベル中の疎らなサブツリーおよび稠密なサブツリーのバイナリツリー表示を示す図である。

【図 16 A】 図 5 に示されるサブツリーエントリおよび図 4 に示されるサブツリーエントリのデータフィールドおよびポインタフィールドに対して変更を加えて、サブツリーエントリ中への複数の疎らなサブツリー記述子の記憶を可能にするのを示す図である。

20

【図 16 B】 図 5 に示されるサブツリーエントリおよび図 4 に示されるサブツリーエントリのデータフィールドおよびポインタフィールドに対する変更を加えてサブツリーエントリ中への複数の疎らなサブツリー記述子の記憶を可能にするのを示す図である。

【図 16 C】 図 5 に示されるサブツリーエントリおよび図 4 に示されるサブツリーエントリのデータフィールドおよびポインタフィールドに対する変更を加えてサブツリーエントリ中への複数の疎らなサブツリー記述子の記憶を可能にするのを示す図である。

【図 17】 疎らに配されたサブツリー中のノードに対してブロックオフセットを選択するための、図 8 に示されたオフセット論理中のスパースモードサブツリー論理を示す図である。

30

【図 18】 図 17 のオフセット論理に示されたスパースモード論理を示す図である。

【図 19 A】 疎らに配されたサブツリー中のノードに対するブロックオフセットの選択を示す図である。

【図 19 B】 疎らに配されたサブツリー中のノードに対するブロックオフセットの選択を示す図である。

【図 19 C】 疎らに配されたサブツリー中のノードに対するブロックオフセットの選択を示す図である。

【図 19 D】 疎らに配されたサブツリー中のノードに対するブロックオフセットの選択を示す図である。

【図 20】 図 8 に示されたポインタ論理中のスパースモードベースセレクト論理を示すブロック図である。

40

【図 21】 サブツリーメモリに記憶される、稠密サブツリー記述子および疎サブツリー記述子を示す図である。

【図 22】 疎らに配されたサブツリーおよび密に配されたサブツリー中のノードに対してルートを記憶するサブツリーマップ中のマップエントリに対してマップアドレスを与えるための方法を図示するフローチャートの図である。

【図 23】 ルックアップテーブルに加えるべき新たなルートのバイナリツリー表示を示す図である。

【図 24】 プロセッサメモリに記憶される更新ルートを示す図である。

【図 25】 ルックアップテーブルに記憶される、図 23 に示された新たなルートを示す

50

図である。

【図 2 6】 図 2 5 に示されたルックアップテーブルに新たなルートを加えるためのステップを図示するフローチャートの図である。

【 図 1 A 】

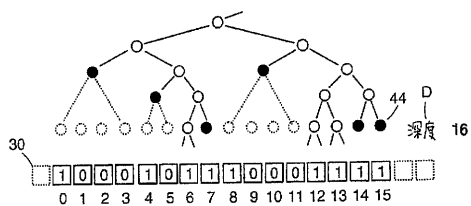


FIG. 1A
PRIOR ART

【 図 1 B 】

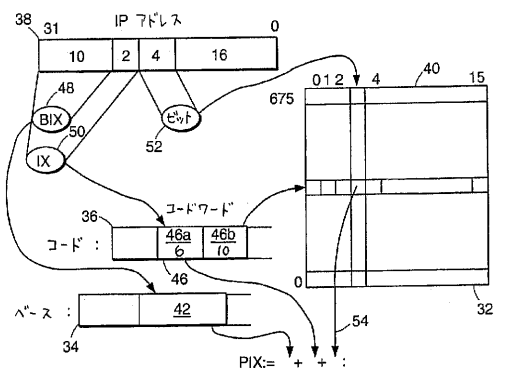


FIG. 1B
PRIOR ART

【圖 2 A】

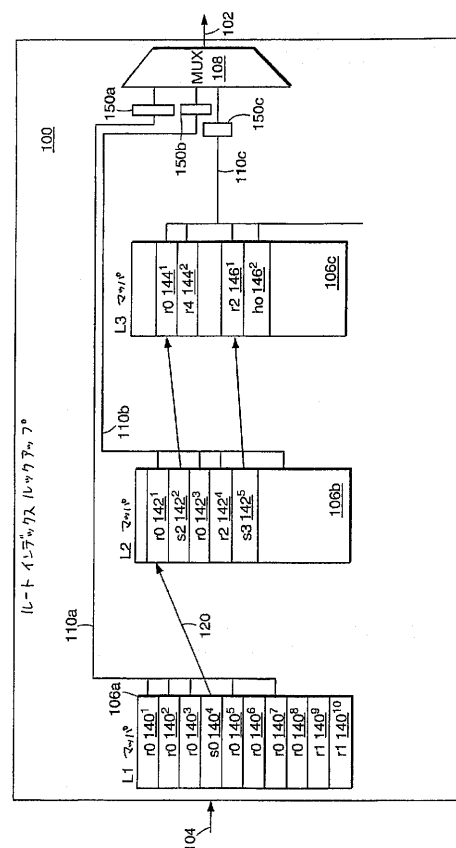


FIG. 2A

【 叉 2 B 】

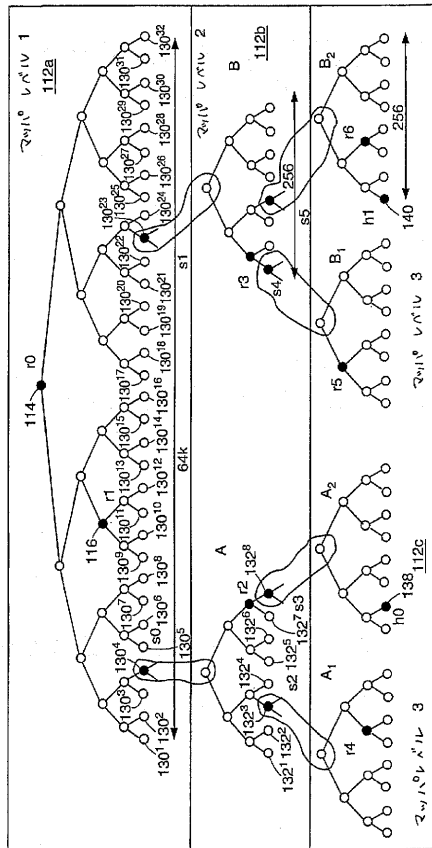


FIG. 2B

【 図 3 】

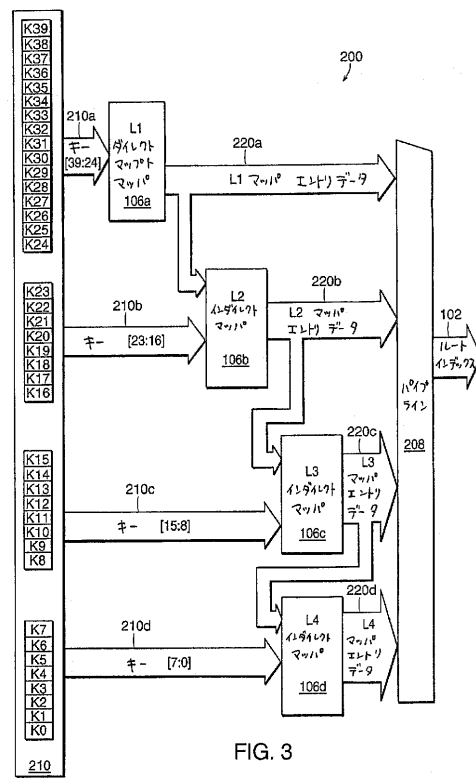


FIG. 3

【 図 4 】

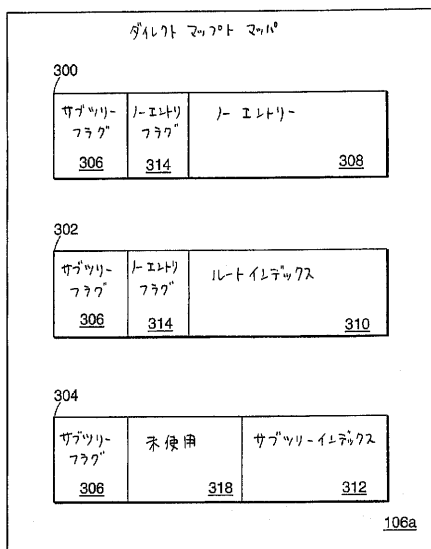


FIG. 4

【 図 5 】

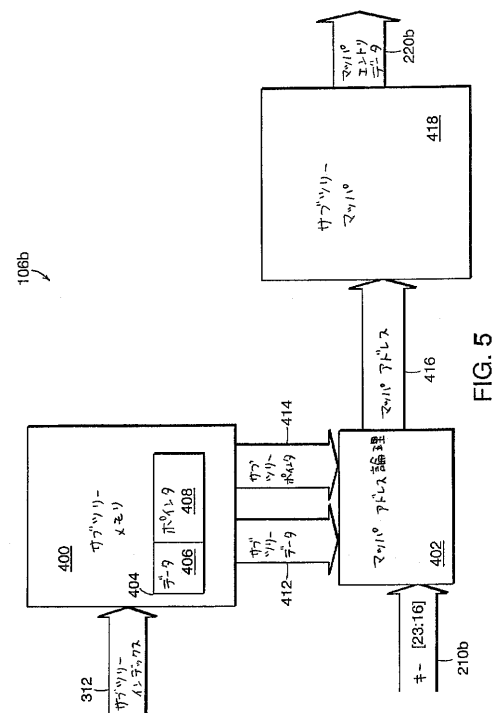


FIG. 5

【図 6 A】

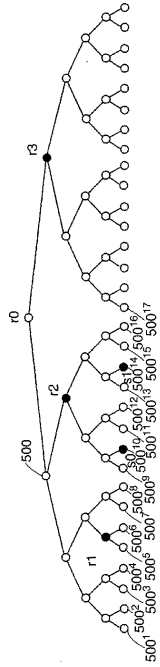


FIG. 6A

【図 6 B】

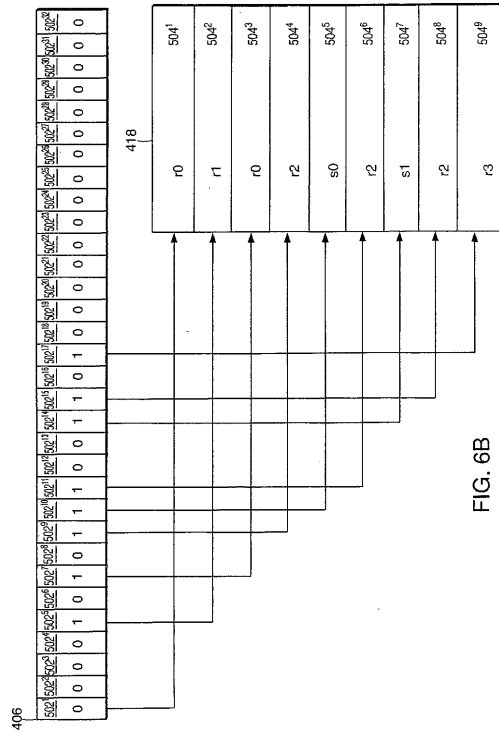


FIG. 6B

【図 7】

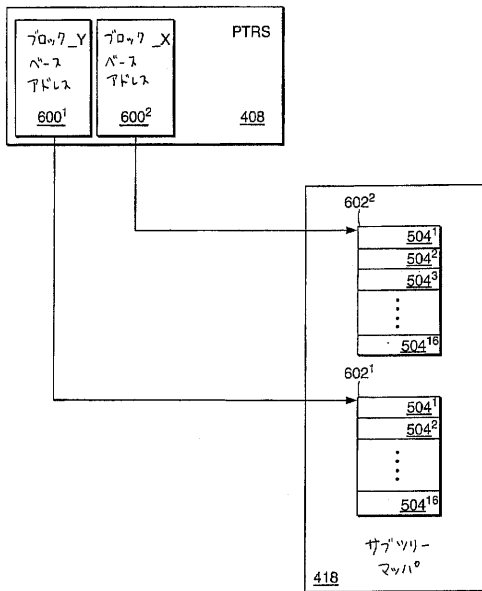


FIG. 7

【図 8】

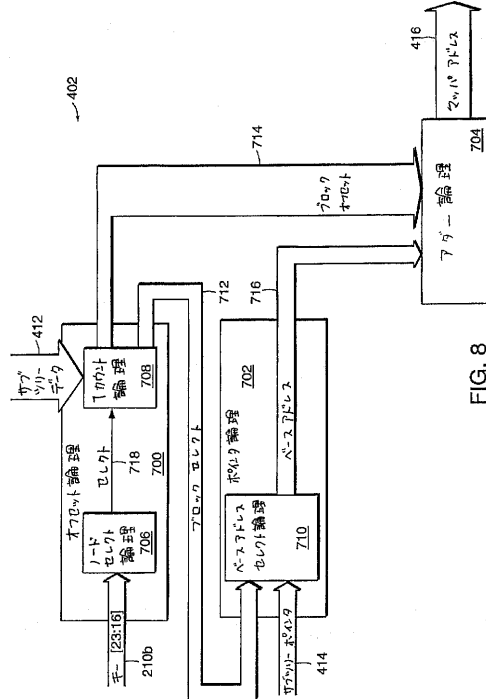


FIG. 8

【図 9】

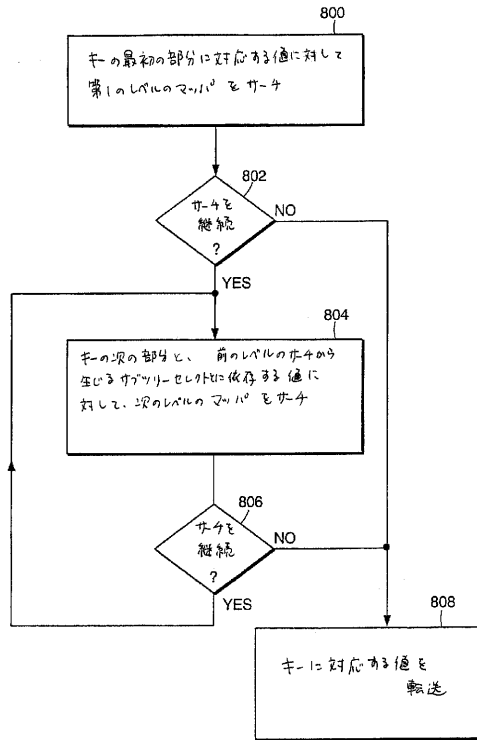


FIG. 9

【図 10 A】

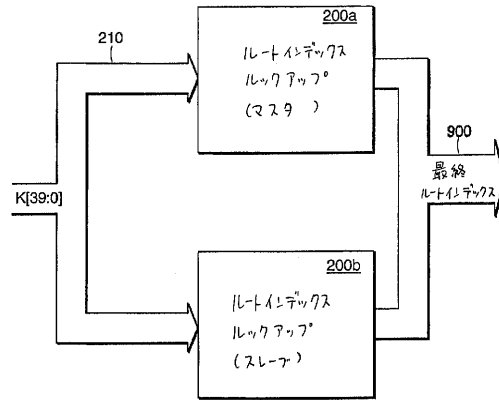


FIG. 10A

【図 10 B】

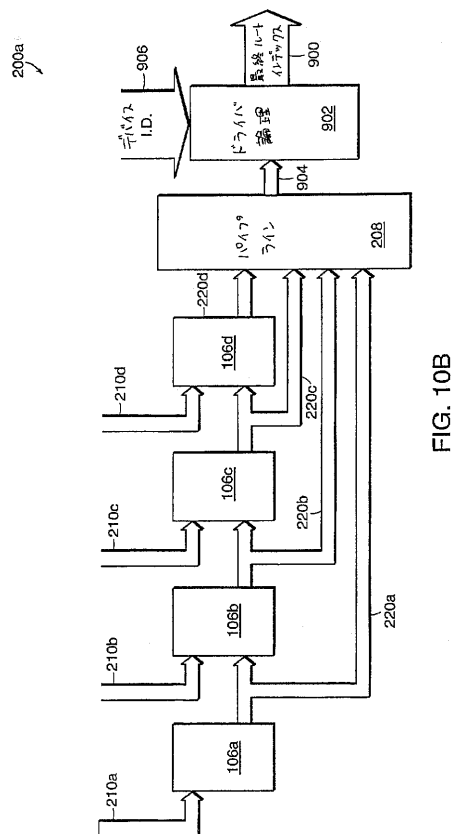


FIG. 10B

【図 10 C】

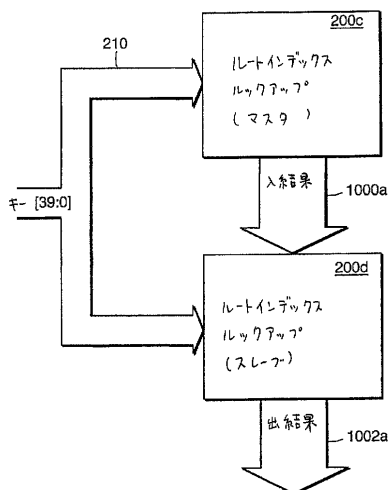


FIG. 10C

【図 10D】

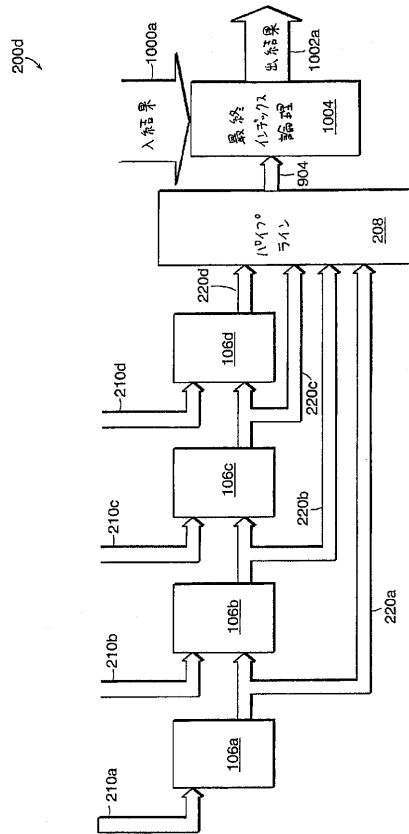


FIG. 10D

【図 11A】

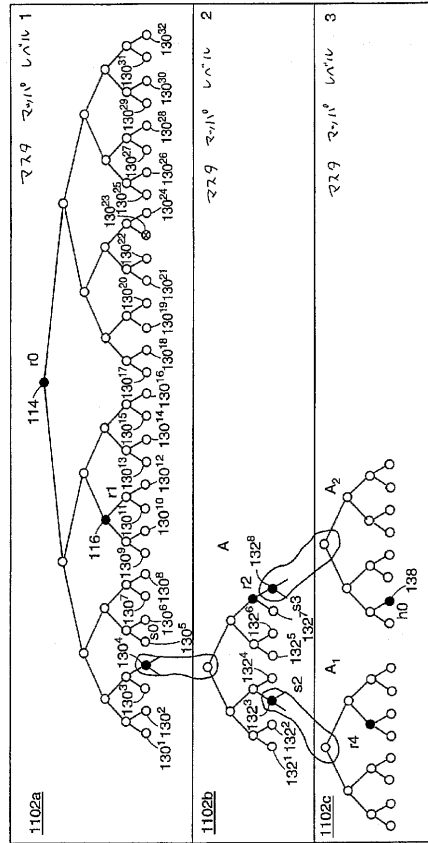


FIG. 11A

【図 11B】

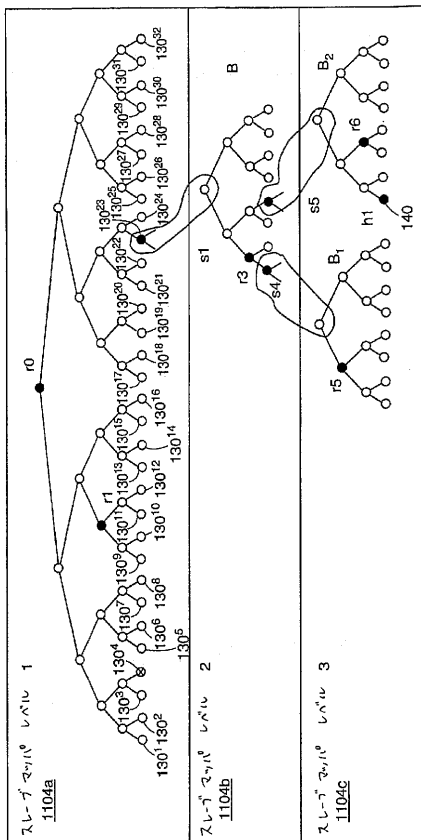


FIG. 11B

【図 12】

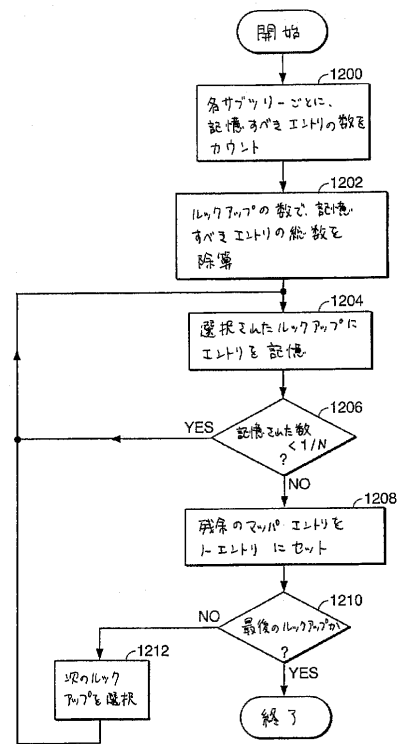


FIG. 12

【図 13】

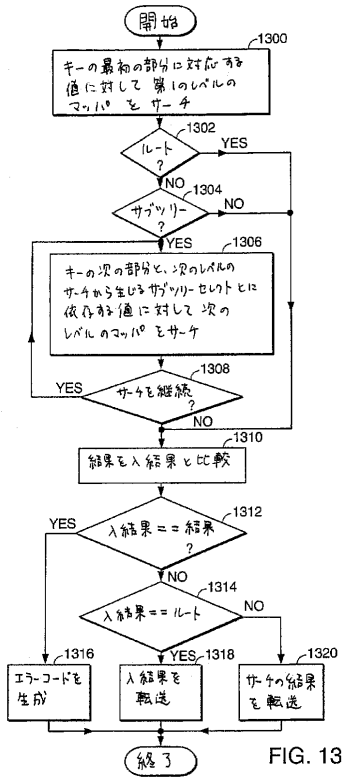


FIG. 13

【図 14】

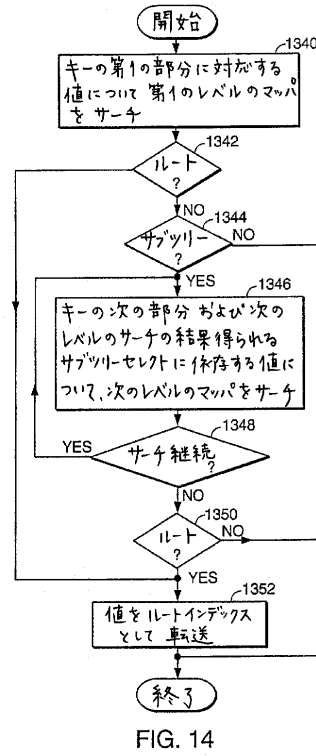


FIG. 14

【図 15】

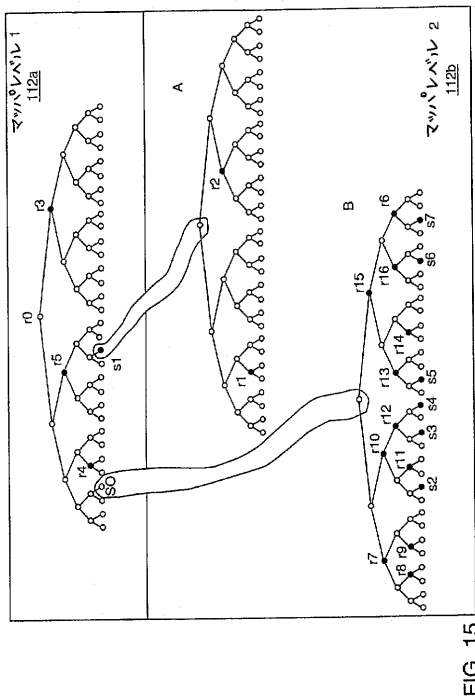


FIG. 15

【図 16 A】

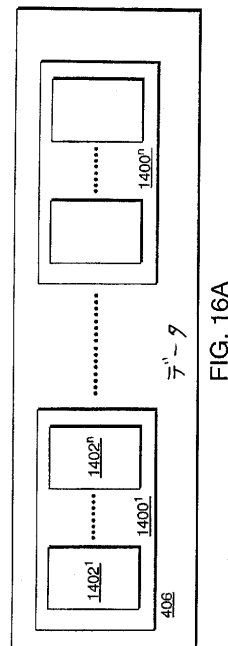


FIG. 16A

【図 16B】

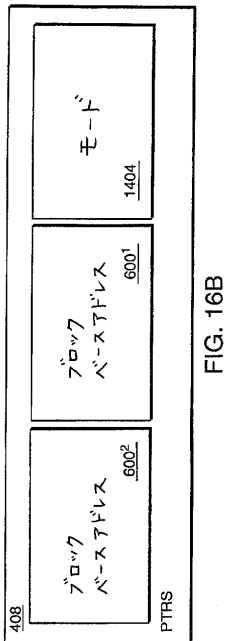


FIG. 16B

【図 16C】

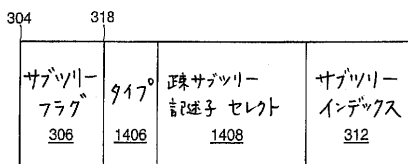


FIG. 16C

【図 18】

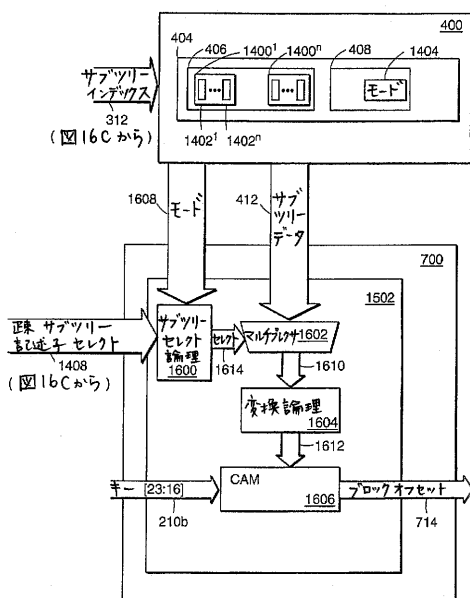


FIG. 18

【図 17】

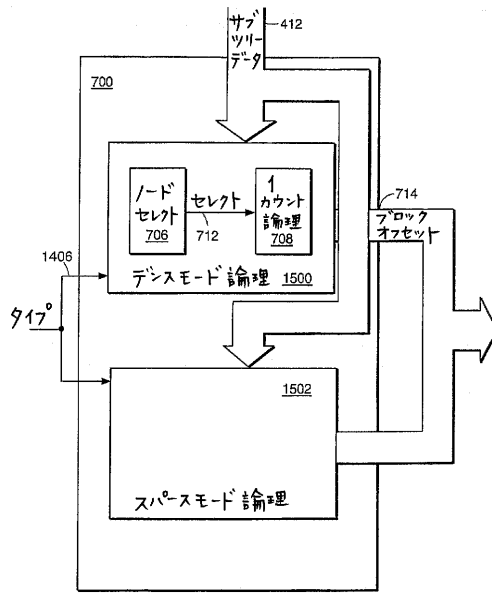
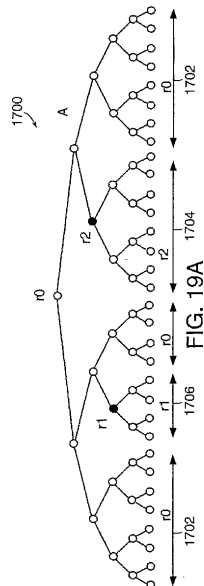
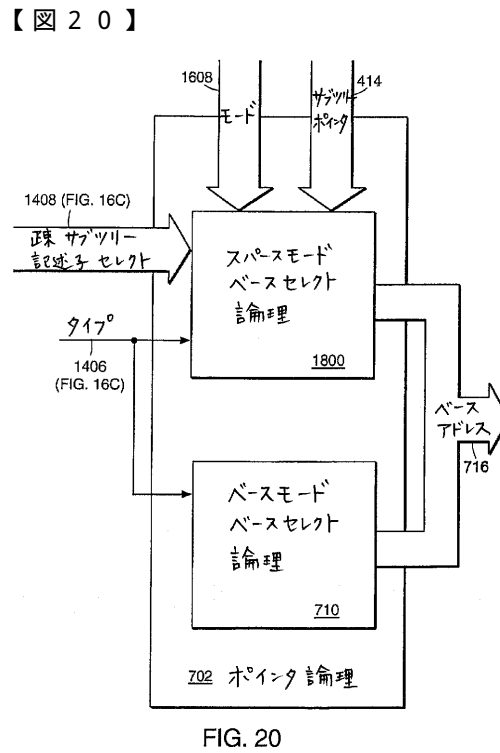
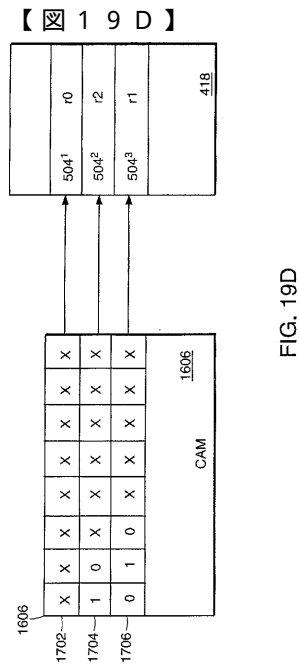
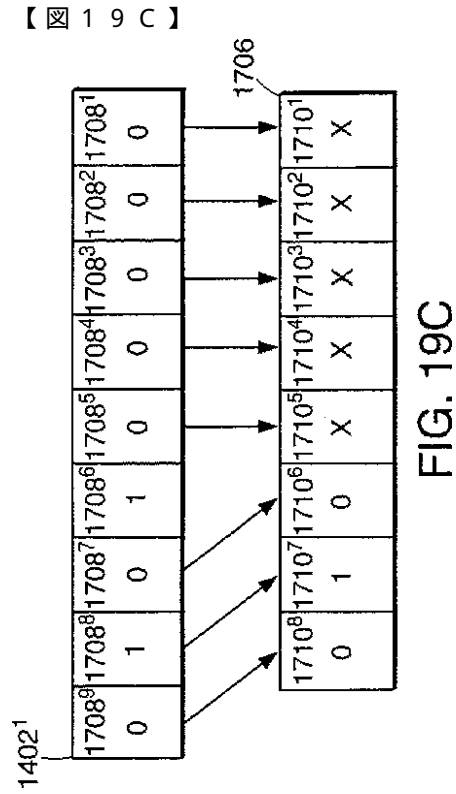
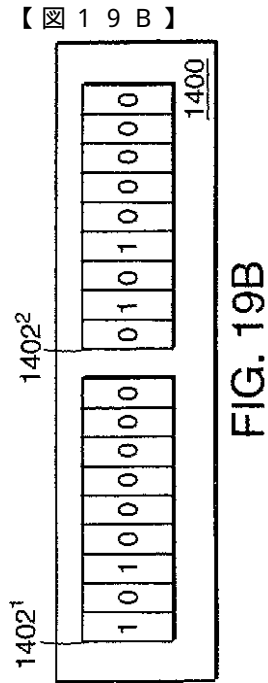


FIG. 17

【図 19A】





【 図 2 1 】

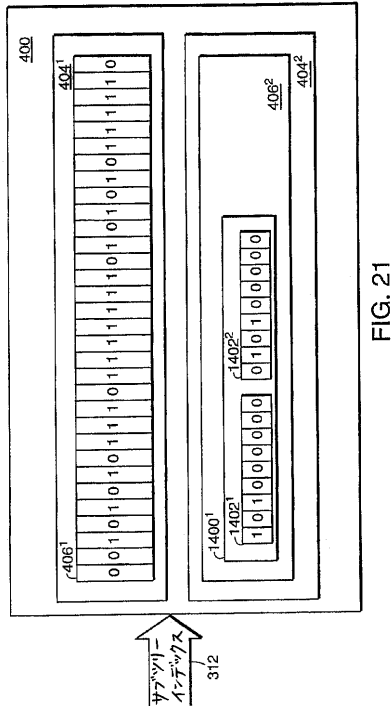


FIG. 21

【 図 2 2 】

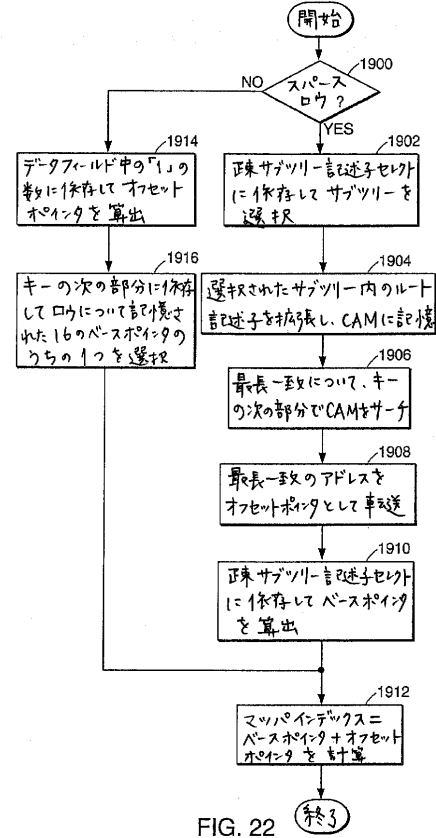


FIG. 22

【 図 2 3 】

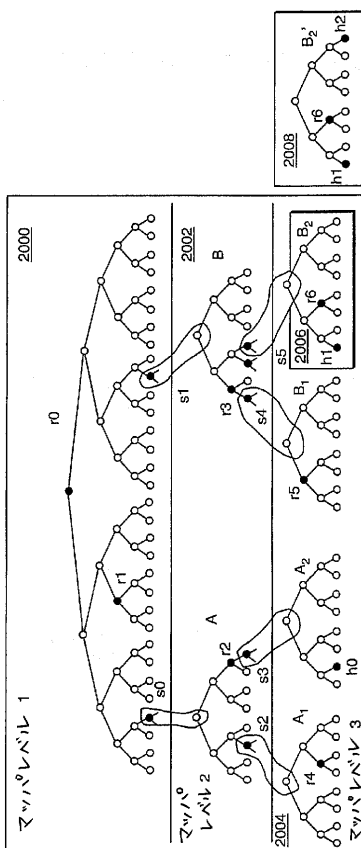


FIG. 23

【 図 2 4 】

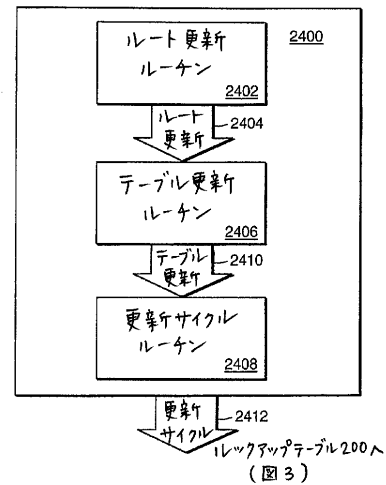


FIG. 24

【図 25】

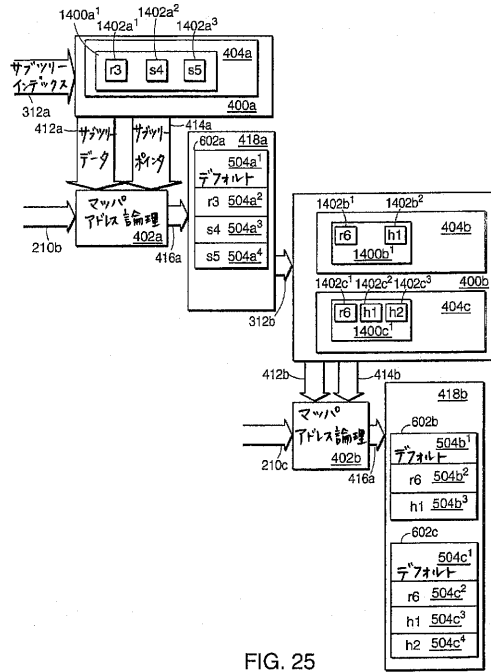


FIG. 25

【図 26】

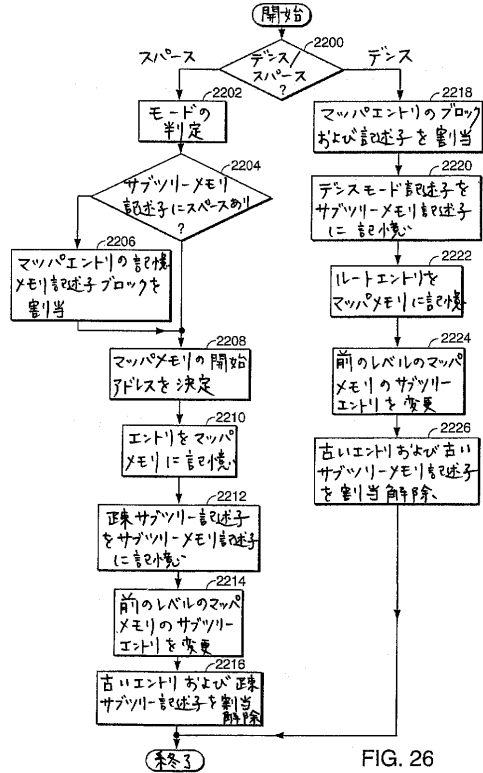


FIG. 26

フロントページの続き

(72)発明者 ブラウン, デイビッド・エイ
カナダ、ケイ・0・エイ 1・エル・0 オンタリオ州、カーブ、アリーシア・クレセント、11
0

審査官 齋藤 浩兵

(56)参考文献 特開平10-257066(JP, A)
国際公開第99/014906(WO, A1)
特開平11-191781(JP, A)
特開平11-066096(JP, A)
特開平11-232285(JP, A)
ポインタキャッシュを用いた最長一致に基づくルーティングテーブル検索法, 信学技報 Vol
. 99 No. 135, 1999年 6月22日, p.7~12
用語解説, 人工知能学会誌 第11巻 第2号, 社団法人人工知能学会, 1996年 3月 1日
, p.337~339

(58)調査した分野(Int.Cl., DB名)
H04L 12/56