(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4892693号
(P4892693)
(45)発行日　平成24年3月7日(2012.3.7)　(24)登録日　平成24年1月6日(2012.1.6)

(51)Int.Cl.　　F I
G11C 15/04 (2006.01)　　G11C 15/04 D
G06F 7/00 (2006.01)　　G11C 15/04 E
G06F 7/00 204

請求項の数 4　(全 43 頁)

(21)出願番号　特願2008-510866 (P2008-510866)
(86)(22)出願日　平成19年3月27日 (2007.3.27)
(86)国際出願番号　PCT/JP2007/056497
(87)国際公開番号　WO2007/119539
(87)国際公開日　平成19年10月25日 (2007.10.25)
審査請求日　平成21年7月3日 (2009.7.3)
(31)優先権主張番号　特願2006-101104 (P2006-101104)
(32)優先日　平成18年3月31日 (2006.3.31)
(33)優先権主張国　日本国(JP)

(73)特許権者　504174135
国立大学法人九州工業大学
福岡県北九州市戸畑区仙水町１番１号
(74)代理人　100121371
弁理士　石田　和人
(72)発明者　笹尾　勤
福岡県飯塚市川津６８０－４　九州工業大学情報工学部内

審査官　堀江　義隆

最終頁に続く

(54)【発明の名称】アドレス生成器

(57)【特許請求の範囲】
【請求項１】
入力される２値ベクトル（以下「入力ベクトル」という。）Ｘに対し、当該入力ベクトルＸが登録ベクトルの場合にはそれに対応する固有アドレスＡを出力し、それ以外の場合には無効値を出力するアドレス生成関数ｆ（Ｘ）の演算を行うアドレス生成器であって、
Ａ．１乃至複数個設けられ、前記入力ベクトルＸが前記登録ベクトル集合の所定の部分集合に属す場合にはそれに対応する固有アドレスＡを出力し、それ以外の場合には無効値を出力する主アドレス生成器と、
Ｂ．すべての前記主アドレス生成器の出力が無効値となり且つ前記登録ベクトル集合に属す前記入力ベクトルＸに対しては、対応する固有アドレスＡを出力し、それ以外には無効値又は入力ベクトルＸに対応する固有アドレスＡを出力する副アドレス生成関数f_2（Ｘ）の演算を行う副アドレス生成器と、
Ｃ．主アドレス生成器又は副アドレス生成器の出力値が無効値以外であればその値を固有アドレスＡとして出力し、それ以外の場合は無効値を出力する出力合成器と、
を備え、
前記各主アドレス生成器は、
ａ．前記入力ベクトルＸの所定の分割（X_{1i}，X_{2i}）（ｉは各主アドレス生成器を識別するインデックス）に対し束縛変数X_{1i}をハッシュ化し、ハッシュ化された束縛変数Y_{1i}を出力するハッシュ回路と、
ｂ．前記束縛変数Y_{1i}に対する割り当てに対し、前記固有アドレスＡが一対一対応す

る場合（ハッシュ衝突が生じない場合）には当該固有アドレスＡを仮アドレスＡ’として出力し、それ以外の場合（ハッシュ衝突が生じる場合）には任意の値又は対応する固有アドレスＡの何れか一つを仮アドレスＡ’として出力する仮アドレス生成器と、

ｃ．アドレス生成関数ｆ（Ｘ）の逆関数であるデータ再生関数f^{-1}（Ａ）の演算を行う演算器であって、前記仮アドレス生成器が出力する仮アドレスＡ’が入力されると、それに対応する再生ベクトルＸ”＝f^{-1}（Ａ’）を出力するデータ再生器と、

ｄ．前記再生ベクトルＸ”と前記入力ベクトルＸとを比較し、両者が一致する場合には前記仮アドレスＡ’を出力し、それ以外の場合には無効値を出力する固有アドレス検出器と、

を備え、

前記副アドレス生成器は、カスケード状に接続された複数の部分関数メモリを備えたＬＵＴカスケード論理回路により構成されており、

前記各部分関数メモリには、前記副アドレス生成関数f_2（Ｘ）を関数分解して得られる複数の部分関数がＬＵＴとして格納されていることを特徴とするアドレス生成器。

【請求項２】

前記副アドレス生成器は、すべての前記主アドレス生成器において前記ハッシュ化された束縛変数$Y_{1\,i}$がハッシュ衝突を生じる入力ベクトルＸに対しては、当該入力ベクトルＸに対応する前記固有アドレスＡを出力し、それ以外の場合には無効値を出力するものであることを特徴とする請求項１記載のアドレス生成器。

【請求項３】

前記仮アドレス生成器は、前記ハッシュ化された束縛変数Y_1においてハッシュ衝突が生じない場合には、当該束縛変数Y_1に対応する固有アドレスＡを前記仮アドレスＡ’として出力する仮アドレス生成関数が、ルックアップ・テーブル（ＬＵＴ）として格納されたハッシュ・メモリであることを特徴とする請求項１記載のアドレス生成器。

【請求項４】

前記データ再生器は、前記データ再生関数f^{-1}（Ａ）が、ＬＵＴとして格納された補助メモリであることを特徴とする請求項１記載のアドレス生成器。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、入力ベクトルに対し対応する固有アドレスを出力するアドレス生成器に関し、特に、書き換えが容易で且つ小面積で実装可能なアドレス生成器に関する。

【背景技術】

【０００２】

ｋ個（ｋは自然数）の異なる２値ベクトルの集合を登録ベクトル集合（set of registered vectors）とする。登録ベクトル集合の各要素と一致する入力に対して１からｋまでの固有アドレス（intrinsic address）に単射し、それ以外の入力に対して０となる関数をアドレス生成関数（address generation function）という。また、アドレス生成関数の演算を行う回路をアドレス生成器（address generator）という。アドレス生成器に入力されるベクトルを、入力ベクトル（input vector）という。入力ベクトルがｎ次元ベクトルであるアドレス生成関数を、ｎ入力のアドレス生成関数という。また、ｎ入力のアドレス生成関数の演算を行う回路をｎ入力のアドレス生成器という。

【０００３】

アドレス生成器は、連想メモリ又は内容検索メモリ（Content Addressable Memory：ＣＡＭ）とも呼ばれ、パターン・マッチング、インターネットのルータ、プロセッサのキャッシュ、ＴＬＢ（Translation Lookaside Buffer）、データ圧縮、データベースのアクセラレータ、ニューラルネット、メモリパッチなど幅広い分野において利用されている。

【０００４】

アドレス生成器の機能をソフトウェアで実現することも可能であるが、ソフトウェアで実現したものは大幅に低速である。そのため、専用のハードウェア（半導体メモリ）を用

いてアドレス生成器を実現することが多い。以下、ハードウェアで構成された従来のアドレス生成器について説明する。

【０００５】

図９は、従来のアドレス生成器（ＣＡＭ）の基本構成の一例を表すブロック図である（特許文献１参照）。アドレス生成器１００は、比較レジスタ１０１、検索ビット線ドライバ１０２、ｋ個のワードW_1～W_k、ｋ個の一致センス回路MSC_1～MSC_k、ｋ個の一致フラグレジスタMFR_1～MFR_k、及びプライオリティ・エンコーダ（優先度付符号化回路）ＰＥを備えている。

【０００６】

比較レジスタ１０１は、ｎビットの入力ベクトルを格納するレジスタである。検索ビット線ドライバ１０２は、比較レジスタ１０１の各ビットを検索ビット線上にドライブする。各ワードW_1～W_kは、それぞれｎビットのＣＡＭセルを備えている。

【０００７】

図１０は、図９のＣＡＭセルの構成回路図である。図１０に例示したＣＡＭセル１０３は、不一致検出型のものである。ＣＡＭセル１０３は、メモリ・セル１０４及び一致比較回路１０５から構成される。メモリ・セル１０４は、１ビットのデータを記憶するＳＲＡＭ構成のメモリ・セルである。図１０においてＤがデータ、ＤＮが反転データを表す。一致比較回路１０５は、メモリ・セル１０４に記憶された１ビットのデータと検索ビット線対ＳＬ，ＳＬＮ上にドライブされる入力ベクトルとを比較し、その一致比較結果を一致線ＭＬ上に出力する。

【０００８】

一致比較回路１０５は、３つのｎＭＯＳトランジスタ（以下「ｎＭＯＳ」という。）１０６，１０７，１０８を備えている。ｎＭＯＳ １０６，１０７は、検索ビット線ＳＬＮと検索ビット線ＳＬとの間に直列に接続されている。ｎＭＯＳ １０６，１０７のゲートは、それぞれ、メモリ・セル１０４のデータＤ，反転データＤＮに接続されている。ｎＭＯＳ １０８は、一致線ＭＬとグランドとの間に接続されている。ｎＭＯＳ １０８のゲートは、ｎＭＯＳ １０６，１０７の間のノード１０９に接続されている。

【０００９】

まず、検索を行う前に、アドレス生成器１００のそれぞれのワードW_1～W_kに、検索対象である登録ベクトルが記憶される。各ワード内の各ＣＡＭセル１０３において、メモリ・セル１０４へのデータの書き込み及びメモリ・セル１０４からのデータの読み出しは、通常のＳＲＡＭと同様にして行われる。

【００１０】

検索時には、まず、比較レジスタ１０１に入力ベクトルが格納される。入力ベクトルの各のビットは、検索ビット線ドライバ１０２により、各々対応する検索ビット線上にドライブされる。

【００１１】

各々のワードW_1～W_kでは、各ＣＡＭセル１０３に予め記憶されている登録ベクトルと検索ビット線上にドライブされた入力ベクトルとの一致検索が同時（並列）に実行され、その結果が一致線ML_1～ML_k上に出力される。これらの検索結果は、それぞれ一致センス回路MSC_1～MSC_kに入力される。各一致センス回路MSC_1～MSC_kは、各検索結果を増幅し、一致センス出力として一致センス出力線MT_1～MT_kに出力する。各一致センス出力は、一致フラグレジスタMFR_1～MFR_kに格納され、一致フラグ出力として一致フラグ出力出力線MF_1～MF_kに出力される。一致フラグは、‘１’が「一致あり」、‘０’が「一致なし」を表すものとする。

【００１２】

各一致フラグ出力は、プライオリティ・エンコーダＰＥに入力される。プライオリティ・エンコーダＰＥでは、所定の優先順位付けに従って、一致が検出されたワードの中から最優先順位のワードのアドレス（最優先一致アドレス：ＨＭＡ）を選択し出力する。各ワードの優先順位は、ワードW_1が最も高く、W_kに向かうに従って順次優先順位が低くな

るものとする。

【００１３】

尚、各ワードW_1～W_k内の各ＣＡＭセル１０３における一致検索は、次のようにして実行される。

【００１４】

まず、初期化動作を実行する。初期化動作では、検索ビット線対ＳＬ，ＳＬＮがともに‘Ｌ’（＝‘０’）とされる。一方、メモリ・セル１０４に記憶されているデータに応じて、一致比較回路１０５のｎＭＯＳ １０６，１０７のうち一方がオン状態、他方がオフ状態となる。従って、ｎＭＯＳ １０６，１０７のうちオン状態の方を介して、両者の間のノード１０９のレベルが‘Ｌ’となり、ｎＭＯＳ １０８はオフ状態となる。この状態で、一致線ＭＬが‘Ｈ’（＝‘１’）状態にプリチャージされる。尚、一致線ＭＬは‘Ｈ’が「一致」を表す。

【００１５】

次に、検索ビット線を介して比較レジスタ１０１に記憶された入力ベクトルの各ビットが各ＣＡＭセル１０３に入力される。これにより、入力ベクトルＳに応じて、検索ビット線対ＳＬ，ＳＬＮの何れか一方が‘Ｈ’、他方が‘Ｌ’となる。

【００１６】

メモリ・セル１０４に記憶されているデータＤと入力ベクトルＳとが一致する場合、ノード１０９のレベルは‘Ｌ’であり、ｎＭＯＳ １０８はオフ状態に保持される。

【００１７】

一方、データＤと入力ベクトルＳとが一致しない場合、ノード１０９のレベルは‘Ｈ’となり、ｎＭＯＳ １０８はオン状態になる。これにより、一致線ＭＬはディスチャージされて‘Ｌ’状態となる。

【００１８】

ｎビットのＣＡＭセル１０３からなるＣＡＭワードの一致線ＭＬは、各ＣＡＭセル１０３のｎＭＯＳ １０８がパラレルに接続されたワイヤードＯＲ回路を構成している。従って、１ワードを構成するｎビットのＣＡＭセル１０３のすべてにおいて一致が検出された場合に限り、一致線ＭＬは‘Ｈ’（「一致」）の状態に保持される。一方、１ビットでもＣＡＭセル１０３で不一致が検出されると、一致線ＭＬは‘Ｌ’（「不一致」）の状態となる。

【００１９】

例えば、検索の結果、一致フラグレジスタMFR_1～MFR_kに、一致フラグとして‘０’，‘１’，‘１’，‘０’，…，‘１’，‘０’が格納されたとする。この場合、ワードW_2，W_3，…，W_{k-1}で一致が検出されている。従って、プライオリティ・エンコーダＰＥは、最も優先順位が高いワードW_2のアドレスをＨＭＡとして出力する。また、一致フラグレジスタMFR_2に格納された一致フラグを‘０’にクリアすることで、その次に優先順位が高いワードW_3のアドレスをＨＭＡとして出力することができる。以下同様にして、一致が検出されたワードのアドレスを順次出力することができる。

【００２０】

図１１は、図９のＣＡＭセルの別の例の構成回路図である。図１１に示すＣＡＭセル１０３’は一致検出型のものであり、図１０と同様、ＳＲＡＭ構成のメモリ・セル１０４及び一致比較回路１０５を備えている。ＣＡＭセル１０３’は、図１０のＣＡＭセル１０３において、一致比較回路１０５のｎＭＯＳ １０８の接続が異なる。図１１のｎＭＯＳ １０８は、一致線ML_aと一致線ML_bとの間に接続されている。ｎＭＯＳ １０８のゲートは、ｎＭＯＳ １０６，１０７の間のノード１０９に接続されている。

【００２１】

ＣＡＭセル１０３’では、検索時には、初期化動作として、ビット線対ＳＬ，ＳＬＮが共に‘Ｈ’とされる。一方、メモリ・セル１０４に記憶されているデータに応じて、一致比較回路１０５のｎＭＯＳ １０６，１０７のうち一方がオン状態、他方がオフ状態となる。従って、ｎＭＯＳ １０６，１０７のうちオン状態の方を介して、両者の間のノード

１０９のレベルが‘Ｈ’となり、ｎＭＯＳ １０８はオン状態となる。この状態で、一致線ＭＬの一端が‘Ｈ’（＝‘１’）状態にプリチャージされる。尚、一致線ＭＬは‘Ｈ’が「不一致」を表す。

【００２２】

ｎビットのＣＡＭセル１０３’からなるＣＡＭワードの一致線ＭＬは、各ＣＡＭセル１０３’のｎＭＯＳ １０８がシリアルに接続されたＡＮＤ回路を構成する。従って、各々のＣＡＭセルの一致線ML_a，ML_bは、各々のＣＡＭセル１０３’のｎＭＯＳ １０８を介して‘Ｈ’にプリチャージされる。

【００２３】

その後、検索ビット線を介して比較レジスタ１０１に記憶された入力ベクトルの各ビットが各ＣＡＭセル１０３’に入力される。これにより、入力ベクトルＳに応じて、検索ビット線対ＳＬ，ＳＬＮの何れか一方が‘Ｈ’、他方が‘Ｌ’となる。

【００２４】

メモリ・セル１０４に記憶されているデータＤと入力ベクトルＳとが一致する場合、ノード１０９のレベルは‘Ｈ’であり、ｎＭＯＳ １０８はオン状態に保持される。

【００２５】

一方、データＤと入力ベクトルＳとが一致しない場合、ノード１０９のレベルは‘Ｌ’となり、ｎＭＯＳ １０８はオフ状態になる。

【００２６】

ＣＡＭワードのｎビットのＣＡＭセル１０３’のすべての状態が確定した後、一致線ＭＬの一方の端部からディスチャージを開始し、他方の端部で一致比較結果を判定する。このとき、１ビットでも不一致のＣＡＭセル１０３’がある場合には、一致比較結果は‘Ｈ’、すなわち、不一致の状態に保持される。一方、すべてのＣＡＭセル１０３’で一致が検出された場合のみ、一致比較結果は‘Ｌ’、すなわち一致状態となる。

【特許文献１】特開２００４－２９５９６７号公報

【特許文献２】特願２００３－３８９２６４号明細書

【特許文献３】特開２００４－２５８７９９号公報

【特許文献４】米国特許第５，０６３，５７３号公報

【非特許文献１】菅野卓雄監修，香山晋編，「超高速デバイス・シリーズ２　超高速ＭＯＳデバイス」，初版，培風館，１９８６年２月，ｐｐ．３２４－３２５．

【非特許文献２】電子情報通信学会編，「ＬＳＩハンドブック」，第１版，オーム社，１９９４年１１月，ｐｐ．５２３－５２５．

【非特許文献３】Kostas Pagiamtzis and Ali Sheikholeslami, "A Low-Power Content-Addressable Memory (CAM) Using Pipelined Hierarchical Search Scheme", IEEE Journal of Solid-State Circuits, Vol.39, No.9, Sept.2004, pp.1512-1519.

【非特許文献４】T.Sasao, M.Matsuura, and Y.Iguchi, "A cascade realization of multi-output function for reconfigurable hardware", International Workshop on Logic and Synthesis (IWLS01), Lake Tahoe, CA, June 12-15, 2001, pp.225-230.

【非特許文献５】T.Sasao and M.Matsuura, "BDD representation for incompletely specified multiple-output logic functions and its applications to functional decomposition," Design Autonmation Conference, June 2005, (pp.373-378).

【発明の開示】

【発明が解決しようとする課題】

【００２７】

ｎ入力のアドレス生成関数は、通常、出力値が非零となる入力ベクトルの数ｋが、ｎ次元ベクトルのすべての組み合わせの数2^nに比べて十分に小さい、いわゆる疎な論理関数（sparse logic function）である。

【００２８】

例えば、ｎ＝１２８，ｋ＝４００００の登録ベクトル集合について考える。この場合、入力ベクトルの取り得る場合の数に対する登録ベクトルの数の割合は、４００００／2^{1}

$2^{8}=1.17549\times10^{-34}$である。この登録ベクトル集合に対するアドレス生成器を実現する最も単純な方法は、真理値表である。しかし、このような真理値表を直接メモリに記憶するのは、メモリが大きくなりすぎるため現実的ではない。

【００２９】

また、上述の従来のアドレス生成器や２段論理回路，ＰＬＡ（Programmable Logic Array）でも実現可能である。しかし、この登録ベクトル集合では、ＬＳＩにした場合、チップ面積や消費電力が大きくなりすぎるという問題がある。また、上述の従来のアドレス生成器は、ＲＡＭに比べると１ビットあたりの価格（ビットコスト）が高価となる。

【００３０】

例えば、上記従来のアドレス生成器は、ＲＡＭに比べると、並列に検索可能であるため高速であるが、デバイスの構成は複雑となる。そのため、このアドレス生成器の１ビットあたりの価格（ビットコスト）は、ＲＡＭに比べる高価なものになる。

【００３１】

また、１ビットあたりの消費電力がＲＡＭに比べて遙かに大きい（非特許文献３参照）。これは、上で説明したように、すべてのＣＡＭセルを同時にアクセスするためである。そのため、１ビットあたりの消費電力は、通常のＲＡＭの数十倍程度にもなる。

【００３２】

そこで、本発明の目的は、検索の高速性を維持しつつも、消費電力を抑え且つデバイスの構造を簡単化して小面積で実装することが可能なアドレス生成器を提供することにある。

【課題を解決するための手段】

【００３３】

本発明の構成を説明する前に、まず、本明細書において使用する用語を定義し、本発明の基本原理について説明する。

【００３４】

〔１〕用語の定義及びいくつかの定理

〔定義１〕（アドレス生成関数）

関数ｆ（Ｘ）：$B^{n}\rightarrow\{0,1,\cdots,k\}$（Ｂ＝｛０，１｝，ｋ∈自然数）において、ｋ個の異なる登録ベクトル$a_{i}\in B^{n}$（ｉ＝１，２，…，ｋ）に対してｆ（a_{i}）＝ｉ（ｉ＝１，２，…，ｋ）が成立し、それ以外の（$2^{n}-k$）個の入力ベクトルa_{i}に対しては、ｆ（a_{i}）＝０が成立するとき、ｆ（Ｘ）を重みｋのアドレス生成関数(address generation function)という。アドレス生成関数は、ｋ個の異なる２値ベクトルに対して、１からｋまでの固有アドレスを生成する。

（定義終わり）

本明細書においては、ｋの値は入力ベクトルの組み合わせ総数2^{n}に比べて十分に小さい（$k<<2^{n}$）と仮定する。

【００３５】

〔定義２〕（登録ベクトル，登録ベクトル表）

ｋ個のｎビットのベクトルの集合を考える。このベクトルの集合を登録ベクトル集合といい、登録ベクトル集合に属する各ベクトルを登録ベクトルという。登録ベクトル集合に属するすべての登録ベクトルに１からｋまでの整数を１対１に対応させる表を登録ベクトル表という。

（定義終わり）

【００３６】

（例１）

（表１）は、４ビットの７個の登録ベクトルからなる登録ベクトル表を示す。この登録ベクトル表に対応するアドレス生成関数を（表２）に示す。いずれも、登録ベクトルに一致した入力ベクトルに対応するアドレスを３ビットの数（例えば、‘００１’）として出力する。入力ベクトルと一致する登録ベクトルが登録ベクトル表の中にない場合には、‘０００’を出力する。

（例終わり）

【００３７】

【表１】

表 1: 登録ベクトル表

固有アドレス	登録ベクトル			
$f_2f_1f_0$	x_4	x_3	x_2	x_1
001	0	0	1	0
010	0	1	1	1
011	1	1	0	1
100	0	1	0	1
101	0	0	1	1
110	1	0	1	1
111	0	0	0	1

【００３８】

【表２】

表 2: アドレス生成関数

x_4	x_3	x_2	x_1	f_2	f_1	f_0
0	0	0	0	0	0	0
0	0	0	1	1	1	1
0	0	1	0	0	0	1
0	0	1	1	1	0	1
0	1	0	0	0	0	0
0	1	0	1	1	0	0
0	1	1	0	0	0	0
0	1	1	1	0	1	0
1	0	0	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	0	0	0
1	0	1	1	1	1	0
1	1	0	0	0	0	0
1	1	0	1	0	1	1
1	1	1	0	0	0	0
1	1	1	1	0	0	0

【００３９】

〔定義３〕（分割）
入力ベクトルをＸ＝（x_1，x_2，…，x_n）とする。Ｘの変数の集合を｛Ｘ｝で表す。｛X_1｝∪｛X_2｝＝｛Ｘ｝且つ｛X_1｝∩｛X_2｝＝φのとき、Ｘ＝（X_1，X_2）をＸの分割（partition）という。ここで、φは空集合を表す。

（定義終わり）

【００４０】
〔定義４〕（分解表，基本分解表，列複雑度）
完全定義関数ｆ（Ｘ）：$B^n \rightarrow B^q$（Ｂ＝｛０，１｝，Ｘ＝（x_1，x_2，…，x_n），ｎ，ｑ∈自然数）が与えられているとする。（X_L，X_H）をＸの分割とする。Ｘの次元（変数の個数）をｄ（Ｘ）と記す。関数ｆ（Ｘ）及びＸの分割Ｘ＝（X_L，X_H）に対して、以下の（１）～（３）の条件を満たす表をｆの分解表という。
（１）2^{n_L}列2^{n_H}行の表である。ここで、n_L＝ｄ（X_L），n_H＝ｄ（X_H）とする。
（２）各行，各列に２進符号のラベルを持ち、列及び行のラベルの集合は、それぞれn_L，n_Hビットのすべてのパターンを要素とする。
（３）表の各要素が、その要素に対応する列及び行のラベルの組み合わせ（X_L，X_H）に対するｆの真理値ｆ（X_L，X_H）である。
X_Lを束縛変数、X_Hを自由変数という。分解表の異なる列パターンの個数を分解表の列複雑度といい、μと記す。分解表の特別な場合として、X_L＝Ｘ，X_H＝φの場合も考える。
また、関数ｆの分解表のうちで、X_L＝（x_1，x_2，…，x_{n_L}）且つX_H＝（x_{n_L+1}，x_{n_L+2}，…，x_n）となるものを基本分解表という。

（定義終わり）

【００４１】
（例２）
（表３）の分解表では、n_L＝３，n_H＝２，μ＝２である。

（例終わり）

【００４２】

【表３】

表 3: 分解表の例

			$\boldsymbol{X}_L$						
			0	0	0	0	1	1	1	1	x_3
			0	0	1	1	0	0	1	1	x_2
			0	1	0	1	0	1	0	1	x_1
	0	0	0	1	0	0	0	1	1	0	
	0	1	1	1	1	1	1	1	1	1	
$\boldsymbol{X}_H$	1	0	1	0	1	1	1	0	0	1	
	1	1	0	1	0	0	0	1	1	0	
	x_5	x_4									

【００４３】

〔定義５〕（Ｃ尺度）

ベクトルＸの変数の順序をＸ＝（ｘ$_1$，ｘ$_2$，…，ｘ$_n$）としたとき、論理関数ｆ（Ｘ）の基本分解表の列複雑度の最大値をｆのＣ尺度という。

（定義終わり）

【００４４】

（例３）

ｆ$_1$＝ｘ$_1$ｘ$_2$∨ｘ$_3$ｘ$_4$∨ｘ$_5$ｘ$_6$のＣ尺度は３であるが、ｆ$_2$＝ｘ$_1$ｘ$_5$∨ｘ$_2$ｘ$_6$∨ｘ$_3$ｘ$_4$のＣ尺度は８である。

（例終わり）

【００４５】

分解表の列複雑度は、ＭＴＢＤＤ（多端子二分決定グラフ）の幅に等しい。従って、論理関数のＣ尺度は、与えられた入力変数の順におけるＭＴＢＤＤの幅の最大値に等しい。与えられた論理関数$f(x_1,x_2,\cdots,x_n)$に対して、Ｃ尺度は容易に計算でき、一意的に定まる。後述するように、Ｃ尺度が小さい関数は、ＬＵＴカスケードで効率的に実現可能である。従って、Ｃ尺度は、論理関数をＬＵＴカスケードで実現する際の複雑さの尺度となる。

【００４６】

〔補題１〕

重みｋの論理関数ｆのＣ尺度は高々ｋ＋１である。

（補題終わり）

【００４７】

〔定理１〕

与えられた関数ｆ（Ｘ）に対して、Ｘ$_L$を分解表の自由変数、Ｘ$_H$を束縛変数とし、μを分解表の列複雑度とする。このとき、関数ｆは図１に示す回路で実現可能である。この場合、２つのブロックＨ，Ｇの間を結ぶ信号線数は高々

【００４８】

【数１】

$$\lceil \log_2 \mu \rceil$$

である。

（定理終わり）

【００４９】

〔定義６〕（関数分解）

一つの関数ｆ（Ｘ）＝ｆ（Ｘ$_L$，Ｘ$_H$）を分解し２つの関数Ｇ，Ｈ（但し、ｆ（Ｘ）＝（Ｇ（Ｈ$_1$（Ｘ$_L$）），Ｘ$_H$））として実現することを関数分解という。関数分解により

得られる関数Ｇ，Ｈを分解の部分関数という。

（定義終わり）

【００５０】

図１において、２つのブロック間を結ぶ信号線数がX_L中の変数の個数よりも少ないとき、関数を実現するためのメモリ量を削減できる可能性がある。与えられた関数を繰り返し関数分解することにより、図２に示すようなＬＵＴカスケードが得られる。

【００５１】

〔定義７〕（ＬＵＴカスケード）

一つの関数ｆ（Ｘ）に対して関数分解を繰り返し行って得られる部分関数をG_1，G_2，…，G_sと記す。

【００５２】

【数２】

$$\begin{aligned} \boldsymbol{R}_1 &= \boldsymbol{G}_1(\boldsymbol{X}_1) \\ \boldsymbol{R}_2 &= \boldsymbol{G}_2(\boldsymbol{X}_2, \boldsymbol{R}_1) \\ \boldsymbol{R}_3 &= \boldsymbol{G}_3(\boldsymbol{X}_3, \boldsymbol{R}_2) \\ &\vdots \\ \boldsymbol{R}_s &= \boldsymbol{G}_s(\boldsymbol{X}_s, \boldsymbol{R}_{s-1}) \\ f(\boldsymbol{X}) &= \boldsymbol{R}_s \\ \boldsymbol{X} &= (\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_s) \end{aligned}$$

各部分関数G_1，G_2，…，G_sをルックアップ・テーブル（ＬＵＴ）で実現し、各ＬＵＴを信号線でカスケード接続したものをＬＵＴカスケードという。各部分関数G_1，G_2，…，G_sを表す各ＬＵＴをセルという。隣接するセル間を接続する信号線をレイルという。

（定義終わり）

【００５３】

〔定理２〕

Ｃ尺度がμの論理関数は、入力数が高々ｑ＋１で、出力数がｑのセルから構成されたＬＵＴカスケードで実現可能である。ここで、

【００５４】

【数３】

$$q = \lceil \log_2 \mu \rceil$$

である。

（定理終わり）

【００５５】

〔定理３〕

関数ｆを実現するＬＵＴカスケードを考える。いまｎを外部入力数、ｓをセル数、ｑを

最大レイル数（セル間の信号線数）、ｐをセルの最大入力数、およびμを関数ｆのＣ尺度とする。

【００５６】

【数４】

$$p \geq \lceil \log_2 \mu \rceil + 1$$

が成立するとき、以下の関係を満たす関数ｆを実現するＬＵＴカスケードが存在する。

【００５７】

【数５】

$$s \leq \left\lceil \frac{n-q}{p-q} \right\rceil$$

（定理終わり）

【００５８】

重みがｋのアドレス生成関数は、レイル数が

【００５９】

【数６】

$$\lceil \log_2(k+1) \rceil$$

のＬＵＴカスケードで実現可能である。しかし、ｋの値が大きいときは、単一のカスケードでは、セルが大きくなり過ぎ、実現困難となる。例えば、ｋ＝４００００のとき、１７入力１６出力のセルが必要となる。この場合、ベクトルの集合をいくつかに分解して、各ベクトルの集合を別々のカスケードで実現すると、必要メモリ量を削減することが可能である。しかし、この場合には、複数のカスケードの出力を纏めるエンコーダが必要となる。

【００６０】

〔２〕本発明の原理

（１）１個のハッシュ・メモリを使用する方法（ハイブリッド法（Hybrid Method））

重みｋのアドレス生成関数ｆ（X_1，X_2）において、入力変数（X_1，X_2）を（Y_1，X_2）と一次変換し、アドレス空間をハッシュ化する。ハッシュ化した後の関数

【００６１】

【数７】

$$\hat{f}(\boldsymbol{Y}_1, \boldsymbol{X}_2) \qquad (1)$$

の分解表（図３）を考える。尚、以下では

【００６２】

【数８】

$$\hat{f}$$

のように「＾」が付された表記をテキスト上では「ｆ＾」のように記す。Y_1の変数（束

縛変数）の個数をｐとする。非零要素が分解表で一様に分布していると仮定すると、２ｐ＞ｋのとき、分解表は、平均すると各列に高々１個の非零要素を持つ。いま、簡単のために、分解表は各列に高々１個の非零要素を持つと仮定する。次に、

【００６３】

【数９】

$$\hat{h}(\boldsymbol{Y}_1)=\begin{cases}0 & (\text{列}\,\boldsymbol{Y}_1\,\text{の要素がすべて}0\,\text{のとき})\\ \text{列}\,\boldsymbol{Y}_1\,\text{での非零要素の最小値} & (\text{列}\,\boldsymbol{Y}_1\,\text{に非零要素が存在するとき})\end{cases} \tag{2a}$$

又は

$$\hat{h}(\boldsymbol{Y}_1)=\max_{\boldsymbol{b}\in B^{n_2}}\hat{f}_1(\boldsymbol{Y}_1,\boldsymbol{b})\quad(n_2\,\text{は}\,X_2\,\text{の要素数}) \tag{2b}$$

とし、ｈ＾（Ｙ$_1$）をハッシュ・メモリ（Hash memory）３’で実現する（図５参照）。

【００６４】

関数ｆ＾の入力は（Ｙ$_1$，Ｘ$_2$）であるのに対し、ｈ＾（Ｙ$_1$）はＹ$_1$の値のみでｆ＾の値を予測するので、ｈ＾（Ｙ$_1$）の値はｆ＾とは異なる可能性がある。そこで、ｈ＾（Ｙ$_1$）の値が正しいか否かを補助メモリ（auxiliary (AUX) memory）４’により検査する。補助メモリ４’の入力数は

【００６５】

【数１０】

$$q=\lceil\log_2(k+1)\rceil \tag{3}$$

で、出力数はｎである（実際には、n-pまで削減できる）。補助メモリ４’の各アドレスに、対応する登録ベクトルを格納しておく。補助メモリ４’の出力と入力の検索パターンが等しい場合には、ハッシュ・メモリ３’は、正しい値を出力しており、ハッシュ・メモリ３’の出力値をそのまま出力する。等しくない場合は、ハッシュ・メモリ３’はｆ＾と異なった値を出力しているので、０を出力する。

【００６６】

ある列に、非零要素が２個以上ある場合（ハッシュ衝突がある場合）には、上記の方法は利用できないため、ｆ＾（Ｙ$_1$，Ｘ$_2$）を

【００６７】

【数１１】

$$\hat{f}(\boldsymbol{Y}_1, \boldsymbol{X}_2) = \hat{f}_1(\boldsymbol{Y}_1, \boldsymbol{X}_2) \vee \hat{f}_2(\boldsymbol{Y}_1, \boldsymbol{X}_2),$$
$$\hat{f}_1(\boldsymbol{Y}_1, \boldsymbol{X}_2) \cdot \hat{f}_2(\boldsymbol{Y}_1, \boldsymbol{X}_2) = 0$$

と関数分解し、関数ｆ＾$_1$（Ｙ$_1$，Ｘ$_2$）の分解表の各列では非零要素が高々１個となるようにする。また、関数ｆ＾$_2$（Ｙ$_1$，Ｘ$_2$）＝ｆ$_2$（Ｘ$_1$，Ｘ$_2$）に関しては、ＬＵＴカスケードや再構成可能ＰＬＡ（例えば、特許文献４参照）等の再構成可能論理回路で実現する。ここで、ｆ$_2$（Ｘ$_1$，Ｘ$_2$）は、ｆ＾$_2$（Ｙ$_1$，Ｘ$_2$）のハッシュ化された束縛変数Ｙ$_1$を束縛変数Ｘ$_1$に変換して得られる関数を表す。

尚、関数ｆ＾$_1$（Ｙ$_1$，Ｘ$_2$）を主アドレス生成関数（master address generation function）という。また、関数ｆ$_2$（Ｘ$_1$，Ｘ$_2$）を副アドレス生成関数（slave address generation function）という。

【００６８】

このアドレス生成器の特徴は、次の通りである。ハッシュ・メモリ３’は、アドレス生成関数を能率よく実現するが、ハッシュ衝突が起こるため、一部の登録ベクトルは表現することができない。一方、ＬＵＴカスケードや再構成可能ＰＬＡ等の再構成可能論理回路は、アドレス生成関数を確実に実現するが、ｋの値が大きい場合には能率が悪い。そこで、図５のアドレス生成器は、これら２つの手法をうまく組み合わせてアドレス生成関数を能率よく実現できるようにしたものである。このアドレス生成器は、ｎの値が大きく、ｋの値が２^{n}に比べて十分に小さい場合に有効である。ハッシュ・メモリ３’と補助メモリ４’で、およそ９０％の登録ベクトルの集合を実現し、残りの１０％の登録ベクトルの集合をＬＵＴカスケードや再構成可能ＰＬＡ等の再構成可能論理回路 ６’で実現する。

【００６９】

（２）２個以上のハッシュ・メモリを使用する方法（スーパー・ハイブリッド法（Super Hybrid Method））

【００７０】

入力変数Ｘに対してＭ個の異なる分割（Ｘ$_{1i}$，Ｘ$_{2i}$）（ｉ＝１，…，Ｍ）を考える。これらの各分割に対して、（１）と同様にアドレス空間をハッシュ化し、ハッシュ・メモリ３’と補助メモリ４’を用いてアドレス生成関数ｆ＾$_{1i}$（Ｙ$_{1i}$，Ｘ$_{2i}$）を実現する（図７参照）。そして、アドレス生成関数ｆ＾（Ｙ$_1$，Ｘ$_2$）を

【００７１】

【数１２】

$$\hat{f}(\boldsymbol{Y}_1, \boldsymbol{X}_2) = \hat{f}_{11}(\boldsymbol{Y}_{11}, \boldsymbol{X}_{21}) \vee \cdots \vee \hat{f}_{1k}(\boldsymbol{Y}_{1k}, \boldsymbol{X}_{2k}) \vee \hat{f}_2(\boldsymbol{Y}_1, \boldsymbol{X}_2),$$
$$\hat{f}_{1i}(\boldsymbol{Y}_{1i}, \boldsymbol{X}_{2i}) \cdot \hat{f}_2(\boldsymbol{Y}_1, \boldsymbol{X}_2) = 0 \quad (i = 1, \cdots, M)$$

と関数分解し、主アドレス生成関数ｆ＾$_{1i}$（Ｙ$_{1i}$，Ｘ$_{2i}$）の分解表の各列では非零要素が高々１個となるようにする。また、副アドレス生成関数ｆ＾$_2$（Ｙ$_1$，Ｘ$_2$）＝ｆ$_2$（Ｘ$_1$，Ｘ$_2$）に関しては、ＬＵＴカスケードや再構成可能ＰＬＡ等の再構成可能論理回路で実現する。ここで、ｆ$_2$（Ｘ$_1$，Ｘ$_2$）は、ｆ＾$_2$（Ｙ$_1$，Ｘ$_2$）のハッシュ化された束縛変数Ｙ$_1$を束縛変数Ｘ$_1$に変換して得られる関数を表す。

【００７２】

一例として、ｋ＝２の場合（図７参照）を考えると、分割（Ｘ$_{11}$，Ｘ$_{21}$），（Ｘ$_{12}$，Ｘ$_{22}$）（図７においては、Ｘ$_{11}$＝Ｘ$_1$，Ｘ$_{21}$＝Ｘ$_2$，Ｘ$_{12}$＝Ｘ$_1$’，Ｘ$_{22}$＝Ｘ$_2$’と記す。）に対して、２つのハッシュ・メモリ３’、３’及び２つの補助メモリ４’、４’を用いて、主アドレス生成関数ｆ＾$_{11}$（Ｙ$_{11}$，Ｘ$_{21}$），ｆ＾$_{12}$（Ｙ$_1$

$_2$，Ｘ$_{22}$）を実現したとする。主アドレス生成関数ｆ＾$_{11}$（Ｙ$_{11}$，Ｘ$_{21}$）は、登録ベクトルのうちの約８０％に対してアドレスＡを返し、ｆ＾$_{12}$（Ｙ$_{12}$，Ｘ$_{22}$）は、登録ベクトルのうち約１６％に対してアドレスＡを返す。Ｙ$_{11}$，Ｙ$_{12}$のハッシュ化が完全にランダムに行われた場合、主アドレス生成関数ｆ＾$_{11}$（Ｙ$_{11}$，Ｘ$_{21}$），ｆ＾$_{12}$（Ｙ$_{12}$，Ｘ$_{22}$）により、すべての登録ベクトルのうちの約９６％の登録ベクトルに対してアドレス生成関数が実現される。したがって、副アドレス生成関数ｆ$_2$（Ｘ$_1$，Ｘ$_2$）は残りの約４％の登録ベクトルに対してアドレス生成関数を実現すればよい。したがって、再構成可能論理回路で実現すべき副アドレス生成関数ｆ$_2$（Ｘ$_1$，Ｘ$_2$）は、ハイブリッド法に比べて更に小さくなり、より小規模な回路により実現することが可能となる。

【００７３】

（例４）

重みがｋのｎ変数のアドレス生成関数ｆ（Ｘ）を考える。

【００７４】

〔ハイブリッド法〕

ハイブリッド法の場合、ハッシュ・メモリ３’の入力数はｐ＝ｑ＋２，出力数はｑである。ここで、ｑは式（３）により表される。また、補助メモリの入力数はｑ，出力数は（ｎ－ｑ－２）である。従って、総メモリ量は、

【００７５】

【数１３】

$$q \cdot 2^{q+2} + (n-q-2) \cdot 2^q = (4n+12q-8) \cdot 2^{q-2}$$

となる。

【００７６】

〔スーパー・ハイブリッド法〕

スーパー・ハイブリッド法の場合、第１のハッシュメモリの入力数は（ｑ＋１），出力数はｑ、第１の補助メモリの入力数はｑ，出力数は（ｎ－ｑ－１）、第２のハッシュメモリの入力数は（ｑ－１），出力数は（ｑ－２）、第２の補助メモリの入力数は（ｑ－２），出力数は（ｎ－ｑ＋２）である。従って、総メモリ量は、

【００７７】

【数１４】

$$q \cdot 2^{q+1} + (n-q-1) \cdot 2^q + (q-2) \cdot 2^{q-1} + (n-q+2) \cdot 2^{q-2} = (5n+5q-6) \cdot 2^{q-2}$$

となる。

【００７８】

上の２つの式より、

【００７９】

【数１５】

$$n \leq 7\log_2(k+1) - 2$$

が成立するとき、スーパー・ハイブリッド法の方が、ハイブリッド方よりも、必要メモリ

量が少なくなる。

（例終わり）

【００８０】

〔３〕本発明の構成

本発明に係るアドレス生成器の第１の構成は、

入力される２値ベクトル（以下「入力ベクトル」という。）Ｘに対し、当該入力ベクトルＸが登録ベクトルの場合にはそれに対応する固有アドレスＡを出力し、それ以外の場合には無効値を出力するアドレス生成関数ｆ（Ｘ）の演算を行うアドレス生成器であって、

Ａ．１乃至複数個設けられ、前記入力ベクトルＸが前記登録ベクトル集合の所定の部分集合に属す場合にはそれに対応する固有アドレスＡを出力し、それ以外の場合には無効値を出力する主アドレス生成器と、

Ｂ．すべての前記主アドレス生成器の出力が無効値となり且つ前記登録ベクトル集合に属す前記入力ベクトルＸに対しては、対応する固有アドレスＡを出力し、それ以外には無効値又は入力ベクトルＸに対応する固有アドレスＡを出力する副アドレス生成関数f_2（Ｘ）の演算を行う副アドレス生成器と、

Ｃ．主アドレス生成器又は副アドレス生成器の出力値が無効値以外であればその値を固有アドレスＡとして出力し、それ以外の場合は無効値を出力する出力合成器と、

を備え、

前記各主アドレス生成器は、

ａ．前記入力ベクトルＸの所定の分割（X_{1i}，X_{2i}）（ｉは各主アドレス生成器を識別するインデックス）に対し束縛変数X_{1i}をハッシュ化し、ハッシュ化された束縛変数Y_{1i}を出力するハッシュ回路と、

ｂ．前記束縛変数Y_{1i}に対する割り当てに対し、前記固有アドレスＡが一対一対応する場合（ハッシュ衝突が生じない場合）には当該固有アドレスＡを仮アドレスＡ’として出力し、それ以外の場合（ハッシュ衝突が生じる場合）には任意の値又は対応する固有アドレスＡの何れか一つを仮アドレスＡ’として出力する仮アドレス生成器と、

ｃ．アドレス生成関数ｆ（Ｘ）の逆関数であるデータ再生関数f^{-1}（Ａ）の演算を行う演算器であって、前記仮アドレス生成器が出力する仮アドレスＡ’が入力されると、それに対応する再生ベクトルＸ”＝f^{-1}（Ａ’）を出力するデータ再生器と、

ｄ．前記再生ベクトルＸ”と前記入力ベクトルＸとを比較し、両者が一致する場合には前記仮アドレスＡ’を出力し、それ以外の場合には無効値を出力する固有アドレス検出器と、

を備えたことを備えたことを特徴とする。

【００８１】

この構成によれば、アドレス生成器に外部から入力ベクトルＸが入力された場合、ハッシュ回路が入力ベクトルＸ＝（X_{1i}，X_{2i}）に対し束縛変数X_{1i}をハッシュ化し、ハッシュ化された束縛変数Y_{1i}として出力する。ここで、Ｘ＝（X_{1i}，X_{2i}）はＸの分割とする。また、ハッシュ化により束縛変数X_{1i}はY_{1i}に変換され、入力ベクトルＸはハッシュ入力ベクトルX_i’＝（Y_{1i}，X_{2i}）となる。

【００８２】

次に、仮アドレス生成器は、Y_{1i}についてハッシュ衝突が生じない場合にはY_{1i}に対応する固有アドレスAを仮アドレスA’として出力する。ハッシュ衝突が生じる場合には任意の値又は対応する固有アドレスAのうちの何れか一つを、仮アドレスA’として出力する。データ再生器は、得られた仮アドレスA’からそのアドレスに対応する再生ベクトルＸ”＝f^{-1}（A’）を出力する。固有アドレス検出器は、Ｘ”とＸを比較し、両者が一致する場合には仮アドレスA’を出力する。それ以外の場合には無効値を出力する。これにより、固有アドレス検出器からは、入力ベクトルＸに対して正しい固有アドレスA又は無効値が出力される。

【００８３】

一方、副アドレス生成器は、固有アドレス検出器の出力が無効値となる入力ベクトルＸ

に対しては、Ｘに対応する正しい固有アドレスＡ＝ｆ（Ｘ）（固有アドレスがない場合は無効値）を出力する。一方、固有アドレス検出器の出力が無効値以外となる場合には、無効値を出力するか、又はＸに対応する正しい固有アドレスＡ＝ｆ（Ｘ）を出力する。

【００８４】

最後に、出力合成器は、固有アドレス検出器の出力値及び副アドレス生成器の出力値のうち無効値でないものがあれば、それを固有アドレスAとして出力する。両者とも無効値の場合は無効値を出力する。これにより、入力ベクトルＸに一致する登録ベクトルがあれば、出力合成器からは、Ｘに対する正しい固有アドレスAが出力される。

【００８５】

入力ベクトルＸを一旦ハッシュ化することにより、アドレス生成関数A＝ｆ（Ｘ）＝ｆ（X_{1i}，X_{2i}）は、A＝ｆ＾（Y_{1i}，X_{2i}）に変換される。固有アドレスAの数ｋがｎ次元の入力ベクトルＸの組み合わせの数2^nに対して十分に小さい場合、Y_{1i}に対応するAの数は平均して高々１個となる。すなわち、ハッシュ関数をうまく選択すれば、Y_{1i}の割り当てとAとはほぼ一対一対応の関係となる。そこで、ハッシュ化の際に使用するハッシュ関数を、Y_{1i}の値を用いて、Aを一意的に決定できる確率をできるだけ大きくなるように選択する。そうすれば、大部分のAは、束縛変数Y_{1i}の関数A＝Ｇ（Y_{1i}）として表される。Ｇの入力ベクトルY_{1i}の要素数が入力ベクトルＸの要素数に比べて少ないため、仮アドレス生成器は小規模な回路で実現することができる。

【００８６】

そして、残りのハッシュ衝突を生じる入力ベクトルＸについては、副アドレス生成器において、AをＸの全要素の関数として計算する。このとき、Ｘはハッシュ衝突を生じるものに限定されているため、副アドレス生成器で計算することが必要なAの組み合わせの数は少ない。従って、副アドレス生成器は小規模な回路により実現することができる。

【００８７】

故に、全体として、アドレス生成器全体の回路規模を小さくすることができ、消費電力を抑えることができる。また、デバイスの構造を簡単化し、小面積で実装することが可能となる。

【００８８】

また、主アドレス生成器を２つ以上備える場合には、各主アドレス生成器での分割（X_{1i}，X_{2i}）を互いに異なる分割とすることで、主アドレス生成器が１つの場合に比べて、主アドレス生成器によりアドレス生成関数が実現される登録ベクトルの割合が大きくなる。したがって、副アドレス生成器によりアドレス生成関数を実現すべき登録ベクトルの数を減らすことができ、より小規模な回路でアドレス生成器を構成することが可能となる。

【００８９】

ここで、「２値ベクトル（binary vector）」とは、ベクトルの各成分が２値であるベクトルをいう。

【００９０】

「ハッシュ化（hashing）」とは、アドレス生成関数の分解表において非零要素が多くの列に分散するように、入力ベクトルＸの一部又は全部のベクトル成分の順序を置換することをいう。「ハッシュ衝突（hash collision）が生じる」とは、ハッシュ入力ベクトルＸ’のベクトル成分からハッシュ化されていないベクトル成分を除いた束縛変数Y_1の割り当てに対し、複数の固有アドレスAが対応することをいう。

【００９１】

「データ再生関数（data-regeneration function）」とは、アドレス生成関数ｆ（Ｘ）の逆関数、すなわち、固有アドレスＡをそれに対応する入力ベクトルＸに写像する関数をいう。

【００９２】

「無効値（invalid value）」とは、アドレスが無効である（存在しない）ことを表す値をいう。無効値としては、アドレスのすべての成分を０としたもの，すべての成分を１

としたもの，実際に存在しないアドレス値としたものなどが使用される。

【００９３】

「副アドレス生成関数（slave address generation function）」とは、前記固有アドレス検出器が無効値を出力する場合には、当該入力ベクトルＸに対しアドレス生成関数ｆ（Ｘ）の演算値を出力し、それ以外の場合には無効値又はアドレス生成関数ｆ（Ｘ）の演算値を出力する関数をいう。すなわち、副アドレス生成関数は、前者の場合には、当該入力ベクトルＸをそれに対応する固有アドレスAに写像し、後者の場合には無効値又は当該入力ベクトルＸに対応する固有アドレスAに写像する。

【００９４】

副アドレス生成器としては、ＬＵＴカスケード、その他の論理回路を使用することができる。

【００９５】

本発明に係るアドレス生成器の第２の構成は、前記第１の構成において、前記副アドレス生成器は、前記主アドレス生成器のいずれかにおいて前記ハッシュ化された束縛変数Y_{1i}がハッシュ衝突を生じる入力ベクトルＸに対しては、当該入力ベクトルＸに対応する前記固有アドレスＡを出力し、それ以外の場合には無効値を出力するものであることを特徴とする。

【００９６】

この構成によれば、副アドレス生成器は、ハッシュ衝突を生ずる場合には正しい固有アドレスを出力し、ハッシュ衝突を生じない場合には、常に無効値を出力する。従って、副アドレス生成器は、ハッシュ衝突が生じる場合のＸを調べ、その結果得られるいくつかのＸに対して固有アドレスAを演算するように構成すればよい。従って、副アドレス生成器における論理演算回路の構成が容易となる。

【００９７】

本発明に係るアドレス生成器の第３の構成は、前記第１の構成において、前記仮アドレス生成器は、前記ハッシュ化された束縛変数Y_1においてハッシュ衝突が生じない場合には、当該束縛変数Y_1に対応する固有アドレスＡを前記仮アドレスＡ’として出力する仮アドレス生成関数が、ルックアップ・テーブル（ＬＵＴ）として格納されたハッシュ・メモリであることを特徴とする。

【００９８】

このように、仮アドレス生成器をハッシュ・メモリで構成することにより、演算速度を高速に維持しつつも、メモリの書き換えによる再構成も簡単に行うことが可能である。

【００９９】

本発明に係るアドレス生成器の第４の構成は、前記第１の構成において、前記データ再生器は、前記データ再生関数f^{-1}（A）が、ＬＵＴとして格納された補助メモリであることを特徴とする。

【０１００】

このように、データ再生器を補助メモリで構成することにより、演算速度を高速に維持しつつも、メモリの書き換えによる再構成も簡単に行うことが可能である。

【０１０１】

本発明に係るアドレス生成器の第５の構成は、前記第１の構成において、前記副アドレス生成器は、カスケード状に接続された複数の部分関数メモリを備えたＬＵＴカスケード論理回路により構成されており、

前記各部分関数メモリには、前記副アドレス生成関数f_2（Ｘ）を関数分解して得られる複数の部分関数がＬＵＴとして格納されていることを特徴とする。

【０１０２】

このように、副アドレス生成器をＬＵＴカスケード論理回路で構成することにより、演算速度を高速に維持しつつも、回路規模も、ＰＬＡ（プログラマブル・ロジック・アレイ）で構成する場合に比べて小規模化することができる。また、メモリの書き換えによる再構成も簡単に行うことが可能である。

【発明の効果】
【０１０３】
以上のように、本発明によれば、入力ベクトルをハッシュするハッシュ回路及びハッシュ入力ベクトルから固有アドレスを出力する仮アドレス生成器を備えるとともに、仮アドレス生成器で計算できない固有アドレスを計算するための副アドレス生成器を備え、仮アドレス生成器及び副アドレス生成器を相補的に組み合わせて固有アドレスの生成を行う構成とした。これにより、アドレス生成器全体の回路規模を小さくすることができ、消費電力を抑え且つデバイスの構造を簡単化して小面積で実装することが可能となる。
【０１０４】
また、仮アドレス生成器やデータ再生器をＬＵＴを格納したメモリで構成し、副アドレス生成器をＬＵＴカスケードにより構成することで、演算速度を高速に維持しつつも、メモリの書き換えによる再構成も簡単に行うことが可能である。
【図面の簡単な説明】
【０１０５】
【図１】論理関数を関数分解により実現した場合を表す図である。
【図２】中間出力を有するＬＵＴカスケードを表す図である。
【図３】ハッシュ化関数ｆ＾（Y_1，X_2）の分解表を表す概念図である。
【図４】本発明の実施例１に係るアドレス生成器の機能構成を表すブロック図である。
【図５】図４のアドレス生成器１の具体的なハードウェア構成を表すブロック図である。
【図６】６変数関数をハッシュ・メモリで実現した例を示す図である。
【図７】本発明の実施例１に係るアドレス生成器のハードウェア構成を表すブロック図である。
【図８】レジスタとゲートを用いた再構成可能ＰＬＡ ６’の構成を表す図である。
【図９】従来のアドレス生成器（ＣＡＭ）の基本構成の一例を表すブロック図である。
【図１０】図６のＣＡＭセルの構成回路図である。
【図１１】図６のＣＡＭセルの別の例の構成回路図である。
【符号の説明】
【０１０６】
１，１’　アドレス生成器
２　ハッシュ回路
３　仮アドレス生成器
３’　ハッシュ・メモリ
４　データ再生器
４’　補助メモリ（AUX memory）
５　固有アドレス検出器
６　副アドレス生成器
６’　ＬＵＴカスケード
６”　再構成可能ＰＬＡ
７　出力合成器
７’　ＯＲ回路
８，８’　主アドレス生成器
１０　比較回路
１１　ＡＮＤ回路
１５　レジスタ
１６　ＥＸＮＯＲゲート
１７　ＡＮＤゲート
【発明を実施するための最良の形態】
【０１０７】
以下、本発明を実施するための最良の形態について、図面を参照しながら説明する。
【実施例１】

【０１０８】
本実施例では、主アドレス生成器が１個の場合（ハイブリッド法）について説明する。
【０１０９】
〔１〕アドレス生成器の構成
図４は、本発明の実施例１に係るアドレス生成器の機能構成を表すブロック図である。本実施例に係るアドレス生成器１は、ハッシュ回路２、仮アドレス生成器３、データ再生器４、固有アドレス検出器５、副アドレス生成器６、及び出力合成器７を備えている。
【０１１０】
アドレス生成器１は、重みｋのアドレス生成関数ｆ（Ｘ）の演算を行う演算器である。すなわち、アドレス生成器１は、外部から入力ベクトルＸが入力されると、入力ベクトルＸに対応する登録ベクトルがある場合にはその登録ベクトルの固有アドレスＡを出力する。それ以外の場合には無効値０を出力する。
【０１１１】

【数１６】

$$f(\boldsymbol{X}) = \begin{cases} A & (\text{if } \boldsymbol{X} \in R) \\ 0 & (\text{otherwise}) \end{cases} \qquad (4)$$

（但し、$A \in \{1, 2, \ldots, k\}$，$R$は登録ベクトルの集合）

【０１１２】
ここで、入力ベクトルＸはｎ次元のベクトルである。また、固有アドレスＡは１～ｋまでの値をとり、ｑビットの２進数で表現されている。ここで、ｑは上述の式（３）により表される。
【０１１３】
ハッシュ回路２は、入力ベクトルＸの一部又は全部を、所定のハッシュ関数に従ってハッシュ化する。ここで、Ｘの分割をＸ＝（$Ｘ_1$，$Ｘ_2$）とする。$Ｘ_1$は束縛変数、$Ｘ_2$は自由変数である。$Ｘ_1$＝Ｘの場合も考える。ハッシュ回路２は、Ｘの束縛変数$Ｘ_1$をハッシュ化しハッシュ入力ベクトルＸ’＝（$Ｙ_1$，$Ｘ_2$）を生成する。
【０１１４】
仮アドレス生成器３は、仮アドレスＡ’を生成する。この場合、仮アドレスＡ’は次のように決定される：
（１）ハッシュ化された束縛変数$Ｙ_1$に対する一つの割り当てが、ハッシュ衝突が生じない場合、その割り当てに対応する固有アドレスＡを仮アドレスＡ’とする；
（２）ハッシュ化された束縛変数$Ｙ_1$に対する一つの割り当てが、ハッシュ衝突が生じる場合、その割り当てに対応する固有アドレスＡの集合｛$Ａ_1$｝の中で最も小さいものを仮アドレスＡ’とする。
【０１１５】
以下、束縛変数$Ｙ_1$に対する一つの割り当て（全部で２^{p}個ある）を仮アドレスＡ’に写像する関数を仮アドレス生成関数と呼び、ｈ＾（$Ｙ_1$）と記す。
【０１１６】
データ再生器４は、アドレス生成関数Ａ＝ｆ（Ｘ）の逆関数$Ｘ＝ｆ^{-1}$（Ａ）を演算す

る演算器である。データ再生器４は、仮アドレス生成器３から入力される仮アドレスＡ’に対して、再生ベクトルＸ”＝ｆ$^{-1}$（Ａ’）を出力する。

【０１１７】

固有アドレス検出器５には、入力ベクトルＸ，再生ベクトルＸ”,及び仮アドレスＡ’が入力される。固有アドレス検出器５は、再生ベクトルＸ”と入力ベクトルＸとの比較を行う。そして、両者が一致する場合は、仮アドレスＡ’を出力し、それ以外は無効値０を出力する。

【０１１８】

副アドレス生成器６は、アドレス生成関数ｆ（Ｘ）に対し、仮アドレス生成器３，データ再生器４，及び固有アドレス検出器５による仮アドレスＡ’の演算と補完関係にある副アドレス生成関数ｆ$_{2}$^（Ｘ）の演算を行い、副アドレスＡ”を出力する演算器である。副アドレス生成関数ｆ$_{2}$^（Ｘ）は、次のような関数である：

（１）固有アドレス検出器５が無効値０を出力する場合には、入力ベクトルＸをそれに対応する固有アドレスＡに写像する；

（２）それ以外の場合には、入力ベクトルＸを無効値０又は対応する固有アドレスＡに写像する。

【０１１９】

副アドレス生成器６は、論理ゲートを組み合わせることによって構成してもよいし、ＬＵＴカスケード論理回路を用いて構成してもよい。

【０１２０】

出力合成器７には、固有アドレス検出器５の出力値、及び副アドレス生成器６の出力値が入力される。出力合成器７は、これらの出力値のうち、一方又は両方が無効値０以外の場合には、その値を固有アドレスＡとして出力する。両方とも無効値０の場合には、無効値０を出力する。

【０１２１】

尚、図４においては、ハッシュ回路２，仮アドレス生成器３，データ再生器４，及び固有アドレス検出器５により、主アドレス生成器８が構成される。

【０１２２】

図５は、図４のアドレス生成器１の具体的なハードウェア構成を表すブロック図である。図４と対応するものには同一符号を付している。図５においては、仮アドレス生成器３は、ハッシュ・メモリ３’によって構成されている。データ再生器４は、補助メモリ４’により構成されている。固有アドレス検出器５は、比較回路１０及びＡＮＤ回路１１により構成されている。副アドレス生成器６は、ＬＵＴカスケード６’により構成されている。また、出力合成器７は、ＯＲ回路７’により構成されている。

【０１２３】

ハッシュ・メモリ３’は書き換え可能メモリにより構成される。ハッシュ・メモリ３’には、式（２ａ）又は式（２ｂ）に示した仮アドレス生成関数ｈ^（Ｙ$_{1}$）がＬＵＴとして記憶されている。

【０１２４】

補助メモリ４’は書き換え可能メモリにより構成される。補助メモリ４’には、アドレス生成関数ｆの逆関数であるデータ再生関数ｆ$^{-1}$がＬＵＴとして記憶されている。

【０１２５】

ＬＵＴカスケード６’は、図２のように複数の信号線（レイル）によってカスケード接続された複数の書き換え可能メモリ（セル）により構成される。ＬＵＴカスケード６’には、副アドレス生成関数ｆ$_{2}$^（Ｘ）がＬＵＴカスケードとして構成されている。尚、本実施例においては、副アドレス生成器６は、ＬＵＴカスケード６’に限られるものではなく、例えば、論理ゲートの組み合わせや再構成可能ＰＬＡ等の他の再構成可能論理回路により構成してもよい。

【０１２６】

〔２〕アドレス生成器の動作

まず、アドレス生成器１にｎ次元ベクトルの入力ベクトルＸが入力される。ハッシュ回路２は、所定のハッシュ関数に従って入力ベクトルＸ＝（X_1，X_2）を（Y_1，X_2）に一次変換し、アドレス空間をハッシュ化する。ここで、（X_1，X_2）はＸの分割であり、X_1は束縛変数、X_2は自由変数である。ベクトルＸの変数の個数をｄ（Ｘ）と記す。ｄ（Ｘ）＝ｎ，ｄ（X_1）＝ｄ（Y_1）＝ｐ，ｄ（X_2）＝ｒ＝ｎ－ｐとする。また、ｑは式（３）で表される。

【０１２７】

ハッシュ回路２において生成されたハッシュ化された束縛変数Ｙ１は、ハッシュ・メモリ３’に入力される。

【０１２８】

ここで、ハッシュ関数は、アドレス生成関数ｆに適応してあらかじめハッシュ回路２に設定されている。このハッシュ関数の生成方法に関しては、後述する。

【０１２９】

ハッシュ化した後のアドレス生成関数をｆ＾（Y_1，X_2）と記す。関数ｆ＾（Y_1，X_2）の分解表は、図３に示したようなものとなる。ハッシュ回路２でのハッシュ化によって、分解表における非零要素は分散される。また、アドレス生成関数ｆの重みｋは、2^nに比べて十分に小さいものとし、束縛変数X_1の個数ｐは、2^pがｋよりも大きいものとする。ハッシュ化により、非零要素が均等に分散されると、図３の分解表の殆どの列は、高々１個の非零要素を持つ。

【０１３０】

ハッシュ・メモリ３’は、束縛変数Y_1に基づき、式（２ａ）又は式（２ｂ）で定義される仮アドレス生成関数ｈ＾（Y_1）により仮アドレスＡ’を生成する。ここで、ｄ（Ａ’）＝ｑである。尚、Y_1の値のみで、Ａの値を予測しているので、Ａ’＝ｈ＾（Y_1）は正しい値Ａ＝ｆ＾（Y_1，X_2）とは異なる値をとる場合がある。

【０１３１】

仮アドレスＡ’は、補助メモリ４’及びＡＮＤ回路１１に入力される。

【０１３２】

補助メモリ４’は、仮アドレスＡ’に対応する再生ベクトルＸ”＝f^{-1}（Ａ’）を生成する。比較回路１０には、再生ベクトルＸ”ともとの入力ベクトルＸとが入力される。

【０１３３】

比較回路１０は、両者を比較し、両者が一致する場合には１、それ以外は０を、ＡＮＤ回路１１に出力する。ＡＮＤ回路１１は、比較回路１０の出力値と仮アドレスＡ’の各ビットとのＡＮＤ演算を行う。この演算結果は、ＯＲ回路７’に出力される。

【０１３４】

これにより、仮アドレスＡ’がＸに対する正しい固有アドレスＡと等しい場合には、ＡＮＤ回路１１からＯＲ回路７’へ仮アドレスＡ’が出力され、それ以外の場合は無効値０が出力される。すなわち、ＡＮＤ回路１１の出力値は、ハッシュ化後のアドレス生成関数ｆ＾（Y_1，X_2）の分解表において、非零要素が２個以上存在する列について最小の非零要素以外のものを０に置き換えた関数となる。以下、この関数を主アドレス生成関数（master address generation function）と呼び、f_1＾（Y_1，X_2）と記す。

【０１３５】

一方、入力ベクトルＸは、ＬＵＴカスケード６’にも入力される。ＬＵＴカスケード６’においては、副アドレス生成関数（slave address generation function）f_2（X_1，X_2）の演算をＬＵＴカスケードにより行い、その結果を副アドレスＡ”としてＯＲ回路７’に出力する。

【０１３６】

尚、副アドレス生成関数f_2（X_1，X_2）は、上述のようにハッシュ化後のアドレス生成関数をｆ＾（Y_1，X_2）を式（５）のように関数分解し、f_2＾（Y_1，X_2）の変数Y_1をハッシュ関数の逆関数によってX_1に変換することによって得られる。

【０１３７】

【数１７】

$$\hat{f}(\boldsymbol{Y}_1,\boldsymbol{X}_2)=\hat{f}_1(\boldsymbol{Y}_1,\boldsymbol{X}_2)\vee\hat{f}_2(\boldsymbol{Y}_1,\boldsymbol{X}_2) \tag{5}$$

従って、ハッシュ化された副アドレス生成関数ｆ$_2$＾（Ｙ$_1$，Ｘ$_2$）は、主アドレス生成関数ｆ$_1$＾（Ｙ$_1$，Ｘ$_2$）が決まれば、一意的に次式（６）により求まる。

【０１３８】

【数１８】

$$\hat{f}_2(\boldsymbol{Y}_1,\boldsymbol{X}_2)=\begin{cases}\hat{f}(\boldsymbol{Y}_1,\boldsymbol{X}_2) & (\text{if } \hat{f}_1(\boldsymbol{Y}_1,\boldsymbol{X}_2)=0),\\ 0 & (\text{otherwise}).\end{cases} \tag{6}$$

【０１３９】

尚、具体的な副アドレス生成関数ｆ$_2$（Ｘ$_1$，Ｘ$_2$）の構成方法に関しては後述する。

【０１４０】

主アドレス生成関数ｆ$_1$＾（Ｙ$_1$，Ｘ$_2$）と副アドレス生成関数ｆ$_2$（Ｘ$_1$，Ｘ$_2$）は、アドレス生成関数ｆ（Ｘ$_1$，Ｘ$_2$）に対して相補的である。すなわち、式（５）より、

【０１４１】

【数１９】

$$f(\boldsymbol{X}_1,\boldsymbol{X}_2)=\hat{f}_1(\boldsymbol{Y}_1,\boldsymbol{X}_2)\vee f_2(\boldsymbol{X}_1,\boldsymbol{X}_2) \tag{7}$$

となる。そこで、ＯＲ回路７’は、ＡＮＤ回路１１の出力値ｆ$_1$＾（Ｙ$_1$，Ｘ$_2$）とＬＵＴカスケード６’の出力値ｆ$_2$（Ｘ$_1$，Ｘ$_2$）とのＯＲ演算を行うことによって、アドレス生成関数ｆ（Ｘ$_1$，Ｘ$_2$）の値を算出する。

【０１４２】

〔３〕ハッシュ関数の構成方法

ハッシュ回路２において、束縛変数Ｘ$_1$をハッシュ化するハッシュ関数は、アドレス生成関数ｆの分解表における非零要素を一様に分布させるものであれば、どのような関数でも利用することができる。ここでは、実現の容易さを考慮して、一例として、次のハッシュ関数を用いる。

【０１４３】

【数２０】

$$\boldsymbol{Y}_1=(y_1,y_2,\ldots,y_p),\quad y_i=x_i\oplus x_j\quad (x_i\in\{\boldsymbol{X}_1\},y_i\in\{\boldsymbol{Y}_1\},x_j\in\{\boldsymbol{X}_2\}) \tag{8}$$

【０１４４】

〔３－１〕ハッシュ関数の生成アルゴリズム

アドレス生成関数ｆ（Ｘ$_1$，Ｘ$_2$）において、束縛変数をＸ$_1$＝（ｘ$_1$，ｘ$_2$，…，ｘ

$_p$）、自由変数をX_2＝（x_{p+1}，x_{p+2}，…，x_n）とする。束縛変数X_1をY_1＝（y_1，y_2，…，y_p）に置換した関数をｆ＾（Y_1，X_2）とする。関数ｆ＾（Y_1，X_2）の分解表の列中で、非零要素を含む列の個数をｗとする。ハッシュ・メモリ３’に多くのアドレスを格納するためには、ｗを最大化するようなY_1を選択すればよい。Y_1は、種々の公知のアルゴリズム（非線形計画法や発見的アルゴリズム）を用いて選択すればよい。

【０１４５】

〔アルゴリズム１〕

X_1の各要素x_i（ｉ＝１，２，…，ｐ）に対して、

【０１４６】

【数２１】

$$y_i = x_i \oplus x'_{j_i}$$

としたとき、ｗが最大となるx_{ji}'∈｛X_2｝を選択する。この操作をｗが増加しなくなるまで繰り返す。

（アルゴリズム終わり）

【０１４７】

このようにして求めたX_j＝（x_{j1}，x_{j2}，…，x_{jp}）によりハッシュ関数を構成する。

【０１４８】

〔３－２〕ハッシュ関数の実現法

重みｋのアドレス生成関数ｆ（X_1，X_2）において、束縛変数X_1＝（x_1，x_2，…，x_p）を

【０１４９】

【数２２】

$$(y_1 \oplus x_{j_1}, y_2 \oplus x_{j_2}, \ldots, y_p \oplus x_{j_p}) \qquad (9)$$

で置き換えて得られる関数をｆ＾（Y_1，X_2）とする。ここで、束縛変数の個数ｐは次式（１０）の条件を満たすものとする。

【０１５０】

【数２３】

$$p \geq \lceil \log_2(k+1) \rceil \qquad (10)$$

【０１５１】

ｐ次元の２値ベクトルａ（∈Y_1）に対してｆ＾（ａ，X_2）の非零出力値が２個以上ある場合、ｆ＾（ａ，X_2）の最小の非零出力値以外の値を０に置き換えた関数を主アドレス生成関数f_1＾（Y_1，X_2）とする。主アドレス生成関数f_1＾（Y_1，X_2）は次式（１１）で表される。

【０１５２】

【数２４】

$$\hat{f}_1(\boldsymbol{Y}_1,\boldsymbol{X}_2)=\begin{cases}\hat{f}(\boldsymbol{Y}_1,\boldsymbol{X}_2) & (\text{if } \hat{f}(\boldsymbol{Y}_1,\boldsymbol{X}_2)=\min\limits_{\boldsymbol{b}\in B^{(n-p)},\hat{f}\neq 0}\hat{f}(\boldsymbol{Y}_1,\boldsymbol{b}))\\ 0 & (\text{otherwise})\end{cases} \tag{11}$$

次に、関数ｆ＿２＾（Ｙ＿１，Ｘ＿２）を式（６）により求める。このとき、式（７）の関係が成り立つ。

【０１５３】

主アドレス生成関数ｆ＿１＾（Ｙ＿１，Ｘ＿２）の分解表の各列において、非零出力値は高々１個しか存在しない。次に、次式（１２）で定義される関数ｈ＾（Ｙ＿１）をハッシュ・メモリ３’で実現する。

【０１５４】

【数２５】

$$\hat{h}(Y_1)=\max_{\boldsymbol{b}\in B^{n-p}}\hat{f}_1(\boldsymbol{Y}_1,\boldsymbol{b}) \tag{12}$$

【０１５５】

但し、関数ｈ＾（Ｙ＿１）の値は主アドレス生成関数ｆ＿１＾（Ｙ＿１，Ｘ＿２）の値と異なっている可能性があるので、補助メモリ４’を用いて出力値が正しいか確認する。また、関数ｆ＿２＾（Ｙ＿１，Ｘ＿２）から次式（１３）の置換を行って副アドレス生成関数ｆ＿２（Ｙ＿１，Ｘ＿２）を生成する。副アドレス生成関数ｆ＿２（Ｙ＿１，Ｘ＿２）は、ＬＵＴカスケード６’で実現される。

【０１５６】

【数２６】

$$x_i=y_i\oplus x_{j_i} \tag{13}$$

【０１５７】

（例５）

重みがｋ＝７の６変数のアドレス生成関数ｆ（Ｘ＿１，Ｘ＿２）の分解表を（表４）に示す。この関数で束縛変数Ｘ＿１＝（ｘ＿１，ｘ＿２，ｘ＿３）を次式（１４）のようにＹ＿１に置換し、ハッシュ化した関数ｆ＾（Ｙ＿１，Ｘ＿２）の分解表を（表５）に示す。

【０１５８】

【数２７】

$$\boldsymbol{Y}_1=(y_1,y_2,y_3)=(x_1\oplus x_6,x_2\oplus x_5,x_3\oplus x_4) \tag{14}$$

【０１５９】

ハッシュ化した関数では、元の関数の列が入れ替わっている。また、入れ替えの方法は、行ごとに異なっている。元の関数では、（ｘ＿１，ｘ＿２，ｘ＿３）＝（０，０，０），（０，１，０），（０，０，１）の列に非零要素が２個存在する。一方、（表５）に示すハッシュ化後の関数ｆ＾（Ｘ＿１，Ｘ＿２）では、（ｙ＿１，ｙ＿２，ｙ＿３）＝（０，１，０）の列にのみ、非零要素が２個存在する。そのうち、大きい方の非零要素の４を０に置き換えた関数ｆ＿１＾（Ｙ＿１，Ｘ＿２）の分解表を（表６）に示す。また、（表７）にＬＵＴカスケード

６’で実現すべき関数f_2^（Y_1，X_2）の分解表を示す。この例の場合、わずか１個の非零要素しかない。残りの関数は、（表８）に示すハッシュ・メモリ３’と（表９）の補助メモリ４’で実現する。ハッシュ・メモリ３’の出力h^（Y_1）は、（y_1，y_2，y_3）のみの値を用いて、関数f_1^の値を予測しており、f_1^とは異なっている可能性がある。そのため、（表９）に示す補助メモリ４’を用いて、その出力が正しいか否かを検証する。ＬＵＴカスケード６’で実現すべき関数は、非零出力値４をとる。その入力時の組み合わせは（x_1，x_2，x_3，x_4，x_5，x_6）＝（０，０，１，０，１，１）である。

【０１６０】

図６に、本方式で、関数ｆを実現する回路の全体図を示す。ＬＵＴカスケード６’の部分は、ＡＮＤゲートのカスケードで実現している。また、非零出力は４であるが、２進表現では（１，０，０）であるので、ＯＲ回路７’においては、ＡＮＤ回路１１の出力ビットのうち、最上位の出力ビットのみ、カスケードの出力とのＯＲをとっている。

（例終わり）

【０１６１】

【表４】

表 4: 関数 $f(X_1, X_2)$ の分解表

	0	0	0	0	1	1	1	1	x_3
	0	0	1	1	0	0	1	1	x_2
	0	1	0	1	0	1	0	1	x_1
000	0	0	0	0	0	0	0	0	
001	0	0	0	0	0	0	0	0	
010	1	0	2	0	3	0	0	0	
011	0	0	0	0	4	0	0	0	
100	5	0	0	0	0	0	0	0	
101	0	0	0	0	0	0	0	0	
110	0	0	0	0	0	0	0	6	
111	0	0	7	0	0	0	0	0	
$x_6x_5x_4$									

【０１６２】

【表５】

表 5: ハッシュ化後の関数 $\hat{f}(Y_1, X_2)$ の分解表

	0	0	0	0	1	1	1	1	y_3
	0	0	1	1	0	0	1	1	y_2
	0	1	0	1	0	1	0	1	y_1
000	0	0	0	0	0	0	0	0	
001	0	0	0	0	0	0	0	0	
010	2	0	1	0	0	0	3	0	
011	0	0	4	0	0	0	0	0	
100	0	5	0	0	0	0	0	0	
101	0	0	0	0	0	0	0	0	
110	0	0	0	0	6	0	0	0	
111	0	0	0	0	0	7	0	0	
$x_6x_5x_4$									

【０１６３】

【表６】

表 6: 関数 $\hat{f}_1(Y_1, X_2)$ の分解表

	0	0	0	0	1	1	1	1	y_3
	0	0	1	1	0	0	1	1	y_2
	0	1	0	1	0	1	0	1	y_1
000	0	0	0	0	0	0	0	0	
001	0	0	0	0	0	0	0	0	
010	2	0	1	0	0	0	3	0	
011	0	0	0	0	0	0	0	0	
100	0	5	0	0	0	0	0	0	
101	0	0	0	0	0	0	0	0	
110	0	0	0	0	6	0	0	0	
111	0	0	0	0	0	7	0	0	
$x_6x_5x_4$									

【０１６４】

【表７】

表 7: 関数 $\hat{f}_2(Y_1, X_2)$ の分解表

	0	0	0	0	1	1	1	1	y_3
	0	0	1	1	0	0	1	1	y_2
	0	1	0	1	0	1	0	1	y_1
000	0	0	0	0	0	0	0	0	
001	0	0	0	0	0	0	0	0	
010	0	0	0	0	0	0	0	0	
011	0	0	4	0	0	0	0	0	
100	0	0	0	0	0	0	0	0	
101	0	0	0	0	0	0	0	0	
110	0	0	0	0	0	0	0	0	
111	0	0	0	0	0	0	0	0	
$x_6x_5x_4$									

【０１６５】

【表８】

表 8: ハッシュメモリで実現する関数 $\hat{h}(Y_1)$

y_3	0	0	0	0	1	1	1	1
y_2	0	0	1	1	0	0	1	1
y_1	0	1	0	1	0	1	0	1
$\hat{h}(Y_1)$	2	5	1	0	6	7	3	0

【０１６６】

【表９】

表 9: 補助メモリの内容

$z_1z_2z_3$	x_1	x_2	x_3	x_4	x_5	x_6
000	0	0	0	0	0	0
001	0	0	0	0	1	0
010	0	1	0	0	1	0
011	0	0	1	0	1	0
100	0	0	0	0	0	0
101	0	0	0	0	0	1
110	1	1	1	0	1	1
111	0	1	0	1	1	1

【０１６７】

尚、ハッシュ関数として式（８）を用いた場合、補助メモリ４’のＸ$_1$の部分の出力及び比較回路１０でのＸ$_1$との比較部分は省略することができる。理由は、以下の通りである。

【０１６８】

ハッシュ・メモリ３’で実現する関数は、ｈ＾（Ｙ$_1$）である。この出力値で、補助メモリ４’の値を参照するが、ハッシュ化したアドレス生成関数ｆ＾の分解表では、元のアドレス生成関数ｆの分解表で行の要素が置換されている。従って、ハッシュ・メモリ３’の出力値が正しいことを確認するためには、Ｘ$_2$の値を調べれば十分である。つまり、補助メモリ４’の出力値とＸ$_2$の値が一致すれば、式（８）の関係より、Ｘ$_1$の値も一致することが保証される。従って、ハッシュ・メモリ３’の出力が正しいことが分かる。また、一致しなければ比較回路１０の出力値は０である。

【０１６９】

例えば、図６の回路の場合、補助メモリ４’の出力は、（ｘ$_4$，ｘ$_5$，ｘ$_6$）があれば十分であり、比較回路１０の入力も（ｘ$_1$，ｘ$_2$，ｘ$_3$）は省略可能である。

【０１７０】

〔４〕ハッシュ・メモリで実現可能な登録ベクトルの割合

次に、本発明に係るアドレス生成器において、回路規模をどの程度小さくすることが可能であるかを評価するため、仮アドレス生成器３（ハッシュ・メモリ３’）で実現可能な登録ベクトルの割合を評価する。

まず、仮アドレス生成器３で実現可能な登録ベクトルの割合に関しては、次の定理が成り立つ。

【０１７１】

〔定理４〕

ｆを重みｋのｎ入力アドレス生成関数とし、非零要素が分解表中に一様に分布していると仮定する。このとき、図４における仮アドレス生成器３で実現可能なアドレスの割合δは、次式（）で表される。ここで、ｐ＝｜Ｙ$_1$｜はｆ＾（Ｙ$_1$，Ｘ$_2$）の分解表の束縛変

数の個数である。

【０１７２】

【数２８】

$$\delta \simeq 1 - \frac{1}{2}\left(\frac{k}{2^p}\right) + \frac{1}{6}\left(\frac{k}{2^p}\right)^2 \tag{15}$$

（定理終わり）

【０１７３】

（証明）

ｋは分解表中の非零要素の総数を示す。α＝ｋ／２^{n}は、分解表中での非零要素の割合を示す。β＝１－αは、分解表中での零要素の割合を示す。ｒ＝ｎ－ｐ＝｜Ｘ$_{2}$｜は、ｆ＾（Ｙ$_{1}$，Ｘ$_{2}$）の分解表の自由変数の個数を表す。このとき、ある列の要素がすべて０となる確率Ｐ$_{0}$、及びある列の要素中に少なくとも１つの非零要素が含まれる確率Ｐ$_{1}$は、次式（１６），（１７）により表される。

【０１７４】

【数２９】

$$P_0 = \beta^{2^r} \tag{16}$$

$$P_1 = 1 - \beta^{2^r} \tag{17}$$

全体では２^{p}個の列が存在し、非零要素数はｋである。非零要素が２個以上ある列に関しては、仮アドレス生成器３で１個だけ非零要素を実現すると仮定すると、仮アドレス生成器３で表現可能なアドレスの割合δは、式（１８）により表される。

【０１７５】

【数３０】

$$\begin{aligned}
\delta &= (1-\beta^{2^r})\cdot\frac{2^p}{k} \qquad (18)\\
&= (1-(1-\alpha)^{2^r})\cdot\frac{2^p}{k}\\
&= \left(1-\left(\sum_{i=0}^{2^r}(-1)^i\binom{2^r}{i}\alpha^i\right)\right)\cdot\frac{2^p}{k}\\
&= \left(\sum_{i=1}^{2^r}(-1)^{i+1}\binom{2^r}{i}\alpha^i\right)\cdot\frac{2^p}{k}
\end{aligned}$$

また、ｓ＝ｎ－ｑとおくと、関係ｒ＝ｎ－ｐより、ｒ－ｓ＝ｑ－ｐとなる。ｓ＞ｒであるから、上式の各項の絶対値は、ｉの増加とともに減少する。次に、δを式（１８）の右辺最後の式の第３項までを用いて近似すると、式（１９）で表される。

【０１７６】

【数３１】

$$\begin{aligned}\delta &\simeq [2^r\alpha - \frac{2^{2r}\alpha^2}{2} + \frac{2^{3r}\alpha^3}{6}] \cdot \frac{2^p}{k} \\ &\simeq [2^r(\frac{k}{2^n}) - \frac{1}{2}2^{2r}(\frac{k}{2^n})^2 + \frac{1}{6}2^{3r}(\frac{k}{2^n})^3] \cdot \frac{2^p}{k} \\ &\simeq 1 - \frac{1}{2}(\frac{k}{2^p}) + \frac{1}{6}(\frac{k}{2^p})^2\end{aligned} \tag{19}$$

（証明終わり）

【０１７７】

例えば、ｋ／２ｐ＝１／４とすると、δ　０．８８５４となる。また、１／４≦ｋ／２ｐ≦１／２の領域では、δは、ｋ／２ｐの単調減少関数となる。

【０１７８】

（例６）

ｎ＝４０，ｋ＝１７３０の場合を考える。このとき、

【０１７９】

【数３２】

$$q = \lceil \log_2(k+1) \rceil = \lceil \log_2(1730+1) \rceil = 11$$

であるから、束縛変数の変数の個数をｐ＝１３とする。

【０１８０】

（１）ＬＵＴカスケードのみで実現した場合

セルの入力数をｐ＝１３とすると、カスケードの段数は、（定理２）より、

【０１８１】

【数３３】

$$\lceil \frac{n-q}{p-q} \rceil = \lceil \frac{40-11}{13-11} \rceil = \lceil \frac{29}{2} \rceil = 15$$

セル１個当たりのメモリ量は、２ｐ×ｑ＝２１３×１１bits。従って、総メモリ量は、２１３×１１×１５＝１，３５１，６８０bitsとなる。

【０１８２】

（２）本発明のアドレス生成器１で実現した場合

ｓ＝ｎ－ｑ＝４０－１１＝２９，ｒ＝ｎ－ｐ＝４０－１３＝２７であるから、（定理４）の近似式の仮定は成立する。（定理４）より、

【０１８３】

【数３４】

$$\delta \simeq 1-\frac{1}{2}(\frac{k}{2^p})+\frac{1}{6}(\frac{k}{2^p})^2=1-\frac{1}{2}(\frac{1730}{2^{13}})+\frac{1}{6}(\frac{1730}{2^{13}})^2\simeq 0.901 \qquad (20)$$

となる。ハッシュ・メモリ３’は、ｐ＝１３入力ｑ＝１１出力、補助メモリ４’は、ｑ＝１１入力ｎ＝４０出力。ＬＵＴカスケード６’は、重みが１７３０×（１－０．９０１）＝１７０のアドレス生成関数を実現する。このとき、ＬＵＴカスケードのセルの出力数は、

【０１８４】

【数３５】

$$\lceil \log_2(170+1) \rceil = 8$$

となる。セルの入力数を１０とすると、実現すべきＬＵＴカスケードの段数は、

【０１８５】

【数３６】

$$\lceil \frac{n-q}{p-q} \rceil = \lceil \frac{40-8}{10-8} \rceil = \lceil \frac{32}{2} \rceil = 16$$

となる。最終段以外のセル１個当たりのメモリ量は、高々2^{10}×８bits。最終段のセルのメモリ量は、高々2^{10}×１１bits。カスケードの総メモリ量は、2^{10}×８×１５＋2^{10}×１１＝１３４，１４４bits。ハッシュ・メモリ３’の容量は、2^{13}×１１＝９０１１１２bits。補助メモリ４’の容量は、2^{11}×４０＝８１９２０bitsとなる。従って、総メモリ量は、３０６，１７６bitsとなる。

従って、この例の場合、本発明のアドレス生成器１を使用した方が、ＬＵＴカスケードのみでアドレス生成関数を構成するよりも総メモリ量は少なくてよい。

（例終わり）

【０１８６】

〔５〕実験結果

アドレス生成回路１の応用例として、頻繁に使用する英単語を表にしたものを考える。ここでは、ベンチマークとして３種類の単語表（単語表１，単語表２，単語表３）を用いた。単語帳中の単語の文字数は、最大１３文字であるが、最初の８文字のみを取り扱う。また、８文字より短い単語は、最後に空白を追加して８文字とした。各英文字を５ビットで表現すると、各英単語は４０ビットで表現することができる。また、単語表１，単語表２，単語表３の中の単語数は、それぞれ、１７３０語，３３６６語，４７０５語である。各表において、各単語に固有のインデックス（自然数）を付加する。このとき、単語のインデックスは、それぞれ、１１bits，１２bits，１３bitsとなる。

【０１８７】

次に、ハッシュ関数を生成する。ハッシュ関数の入力数は、（単語帳のインデックスのビット数）＋２とする。単語表１の場合、単語数はｋ＝１７３０、インデックスのビット数は

【０１８８】

【数３７】

$$q = \lceil \log_2(1+k) \rceil = \lceil \log_2(1+1730) \rceil = 11 \qquad (21)$$

束縛変数の個数は、ｐ＝ｑ＋２＝１３。分解表の列の個数は２ｐ＝２１３＝８１９２。非零要素を１個含む列の数は１３８９。非零要素を２個以上含む列の数は、１６５。ハッシュ・メモリ３’で実現不可能な登録ベクトル数は、１７６である。つまり、全単語の９０％がハッシュ・メモリ３’で実現可能であり、残りの１０％をＬＵＴカスケードで実現する必要がある。実験結果を（表１０）に示す。

【０１８９】

【表１０】

表 10: 英単語表の実現

	単語表 1	単語表 2	単語表 3
語数 ：k	1730	3366	4705
入力数 ：n	40	40	40
出力数 ：q	11	12	13
ハッシュ関数の入力数：p	13	14	15
非零要素数が１の列数	1389	2752	4000
非零要素数が２以上の列数	165	293	342
ハッシュメモリで実現できない単語数	176	321	363

【０１９０】

次に、疑似乱数を用いて同等のアドレス生成関数を実現した場合（１００個の平均）の実験結果を（表１１）に示す。この場合、非零要素を１個含む列の数は１３９８．４、非零要素を２個以上含む列の数は１６０．０、ハッシュ・メモリ３’で実現可能な登録ベクトル数は１７１．６であった。同様の実験を単語帳２，３に関しても行った。

【０１９１】

【表１１】

表 11: 擬似乱数で生成したアドレス生成関数の実現

	関数１	関数２	関数３
語数：k	1730	3366	4705
入力数：n	40	40	40
出力数：q	11	12	13
ハッシュ関数の入力数：p	13	14	15
非零要素数が１の列数	1398.4	2737.7	4075.1
非零要素数が２以上の列数	160.0	302.7	307.0
ハッシュメモリで実現できない単語数	171.6	325.6	322.9
定理４から求めた値	169.8	322.1	321.6

【０１９２】

実際の単語表の結果、及び疑似乱数で生成したアドレス生成関数に対する結果は、（定理４）で理論的に求めた結果と大差はない。従って、上記（アルゴリズム１）を用いて生成したハッシュ関数は、非零要素をランダムに分布させる関数として、有効であることが分かる。

【０１９３】

また、必要メモリ量を比較した結果を（表１２）に示す。（表１２）から明らかなように、必要な総メモリ量は、１つのＬＵＴカスケードで実現する場合に比べて、大幅に少ない。

【０１９４】

【表１２】

表 12: メモリ量の比較 (英単語帳)

		単語表１	単語表２	単語表３
カスケード一個で実現				
入力数	n	40	40	40
出力数	q	11	12	13
セルの入力数	p	13	14	15
段数	s	15	14	14
メモリ量	$sq2^p$	1,351,680	2,752,512	5,963,776
ハッシュメモリとカスケードで実現				
ハッシュメモリ量	$q2^p$	90,112	196,608	425,984
補助メモリ量	$n2^q$	81,920	163,840	327,680
カスケードのメモリ量		134,144	301,056	304,104
合計		306,176	661,504	1,056,768

【実施例２】

【０１９５】

本実施例では、主アドレス生成器が２個の場合（スーパー・ハイブリッド法）について説明する。

【０１９６】

図７は、本発明の実施例１に係るアドレス生成器のハードウェア構成を表すブロック図である。図７において、図５と同様の構成部分については同一の符号が付してある。

【０１９７】

図５と比較すると、実施例１のアドレス生成器１は主アドレス生成器８が１つであったのに対し、本実施例のアドレス生成器１’は主アドレス生成器８，８’を２つ備えている点で異なる。

【０１９８】

主アドレス生成器８では、入力ベクトルＸに対する分割（Ｘ$_1$，Ｘ$_2$）について、主アドレス生成関数ｆ＾$_1$（Ｙ$_1$，Ｘ$_2$）の演算を行う。ここで、Ｙ$_1$はＸ$_1$をハッシュ化したベクトルであり、式（８）と同様に計算される。また、主アドレス生成器８’では、入力ベクトルＸに対する分割（Ｘ$_1$’，Ｘ$_2$’）について、主アドレス生成関数ｆ＾$_1$’（Ｙ$_1$’，Ｘ$_2$’）の演算を行う。ここで、Ｙ$_1$’はＸ$_1$’をハッシュ化したベクトルであり、式（８）と同様に計算される。

【０１９９】

ここで、分割（Ｘ$_1$，Ｘ$_2$）と分割（Ｘ$_1$’，Ｘ$_2$’）とは異なる分割とする。具体的には、最適な分割（Ｘ$_1$’，Ｘ$_2$’）については次のような手順により決定すればよい。

【０２００】

（１）アドレス生成関数ｆに対して、

（１－１）第１のハッシュメモリの入力数は（ｑ＋１），出力数はｑ、第１の補助メモリの入力数はｑ，出力数は（ｎ－ｑ－１）となるように、変数を順に分割する。

（１－２）なるべく、分解表の非零要素が分散するようにハッシュ関数を求める。この構成で実現された固有アドレスが表現する関数をｆ＾$_{1\,1}$とする（ランダム関数の場合は元の固有アドレスの約８０％が表現できる）。

（２）上記操作で実現されない固有アドレスが表現する関数、つまりｆ－ｆ＾$_{1\,1}$に対して、

（２－１）第２のハッシュメモリの入力数は（ｑ－１），出力数は（ｑ－２）、第２の

補助メモリの入力数は（ｑ－２），出力数は（ｎ－ｑ＋２）となるように、変数を順に分割する。

（２－２）なるべく、分解表の非零要素が分散するようにハッシュ関数を求める。この構成で実現された固有アドレスが表現する関数をｆ＾$_{12}$とする（ランダム関数の場合は元の固有アドレスの約１６％が表現できる）。

（３）上記何れの操作でも実現されない固有アドレスが表現する関数に対しては、ＬＵＴカスケード又は再構成可能ＰＬＡなどの方法で実現する。

【０２０１】

このように、主アドレス生成器８，８’を２つ備えたことで、実施例１の場合に比べると、主アドレス生成器８，８’により固有アドレスが決定される登録ベクトルの割合が多くなる。したがって、副アドレス生成関数ｆ＾$_{2}$（Ｘ$_{1}$，Ｘ$_{2}$）により固有アドレスが決定される登録ベクトルの数は少なくなるため、副アドレス生成器６（再構成可能ＰＬＡ６”）の回路規模をより小さくすることが可能となる。

【０２０２】

（例７）

重みｋ＝１７３０で変数の数がｎ＝４０のアドレス生成関数ｆ（Ｘ）を、図７のアドレス生成器１’を用いて実現する場合を考える。尚、比較のため、図５のアドレス生成器１についても考察する。

この場合、

【０２０３】

【数３８】

$$q = \lceil \log_2(k+1) \rceil = \lceil \log_2(1+1730) \rceil = 11 \qquad (22)$$

である。

【０２０４】

〔再構成可能ＰＬＡ〕

再構成可能ＰＬＡ ６”を実現する方法としては、種々の方法があるが、ここでは、入力ベクトルの各ビットの値を記憶するレジスタとＡＮＤゲートを使用したレジスタ-ゲート・アプローチ（register and gates approach）を採用する。図８は、レジスタとゲートを用いた再構成可能ＰＬＡ ６”の構成を表す図である。図８において、再構成可能ＰＬＡ ６”は、ｎ個のレジスタ１５、ｎ個のＥＸＮＯＲゲート１６、及び１個のＡＮＤゲート１７により構成されている。

【０２０５】

このアプローチでは、どのような幅のワードであっても構成することができ、高速に再構成することが可能である。入力数をｎ、出力数をｑ、実現すべきアドレス生成関数の重みをｋとすると、図８の再構成可能ＰＬＡ ６”をＦＰＧＡで構成する場合に必要なＬＲ（Logic Element）の数は、

【０２０６】

【数３９】

$$\left(\left\lceil \frac{n}{2} \right\rceil + \left\lceil \frac{2n-1}{3} \right\rceil \right) k + \left\lceil \frac{\frac{k}{2}-1}{3} \right\rceil q \qquad (23)$$

となる。

【０２０７】

この例では、再構成可能ＰＬＡ ６”において実現すべき登録ベクトルの数は１７３０である。従って、式（２３）より、再構成可能ＰＬＡ ６”を実装するのに必要なＬＥ数は、８４，４７８となる。

【０２０８】

〔ハイブリッド法〕

仮アドレス生成器３で表現可能なアドレスの割合δは、式（１９）より、

【０２０９】

【数４０】

$$\begin{aligned}\delta &\simeq 1-\frac{1}{2}\left(\frac{k}{2^p}\right)+\frac{1}{6}\left(\frac{k}{2^p}\right)^2 \\ &= 1-\frac{1}{2}\left(\frac{1730}{2^{13}}\right)+\frac{1}{6}\left(\frac{1730}{2^{13}}\right)^2 \simeq 0.901\end{aligned} \tag{24}$$

となる。図５のハッシュ・メモリ３’の入力数はｐ＝１３、出力数はｑ＝１１である。補助メモリ４’の入力数はｑ＝１１、出力数はｒ＝ｎ－ｐ＝２７である。ハッシュ・メモリ３’のサイズは$2^{13}\times 11=90,112$（bits）である。補助メモリ４’のサイズは$2^{11}\times 27=55,297$（bits）である。故に、メモリ・サイズの総量は、１４５，４０８（bits）である。再構成可能ＰＬＡ ６”で実現される登録ベクトルの数は１７３である。

【０２１０】

〔スーパー・ハイブリッド法〕

仮アドレス生成器３により表現可能なアドレスの割合δは、式（１９）より、

【０２１１】

【数４１】

$$\begin{aligned}\delta &\simeq 1-\frac{1}{2}\left(\frac{k}{2^p}\right)+\frac{1}{6}\left(\frac{k}{2^p}\right)^2 \\ &= 1-\frac{1}{2}\left(\frac{1730}{2^{12}}\right)+\frac{1}{6}\left(\frac{1730}{2^{12}}\right)^2 \simeq 0.8185\end{aligned} \tag{25}$$

となる。図７の第１のハッシュ・メモリ３’の入力数は$p_1=12$、出力数は$q_1=11$である。第１の補助メモリ４’の入力数は$q_1=11$、出力数は$r_1=n-p_1=27$である。第２のハッシュ・メモリ３’の入力数は$p_2=10$、出力数は$q_2=9$である。第２の補助メモリ４’の入力数は$q_2=9$、出力数は$r_2=n-p_2=30$である。

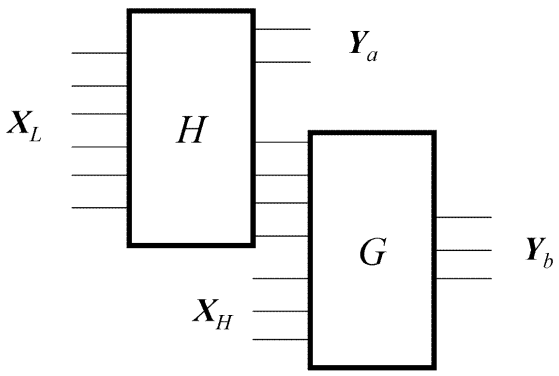
【０２１２】

第１のハッシュ・メモリ３’のサイズは、$2^{12}\times 11=45,056$（bits）、第１の補助メモリ４’のサイズは$2^{11}\times 28=57,344$（bits）、第２のハッシュ・メモリ３’のサイズは、$2^{10}\times 9=9,216$（bits）、第２の補助メモリ４’のサイズは$2^9\times 30=15,360$（bits）である。故に、メモリ・サイズの総量は、１２６，

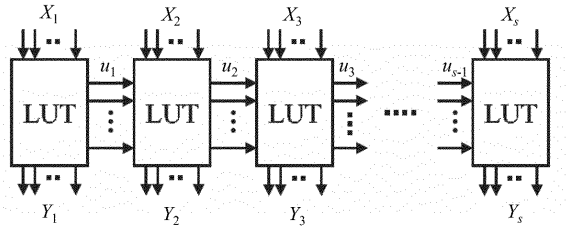
９７６（bits）である。再構成可能ＰＬＡ ６”で実現される登録ベクトルの数は４３である。従って、この例では、スーパー・ハイブリッド法の方がハードウェア量をより少なくすることができる。

（例終わり）

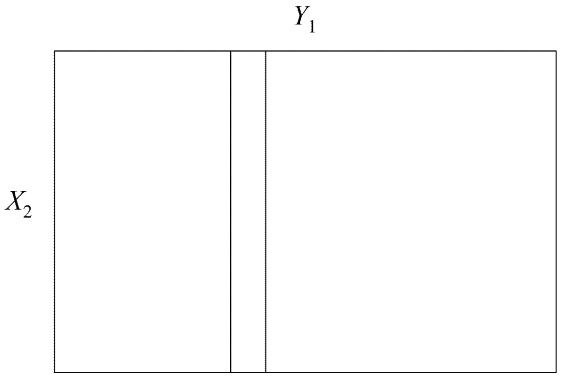
【図１】



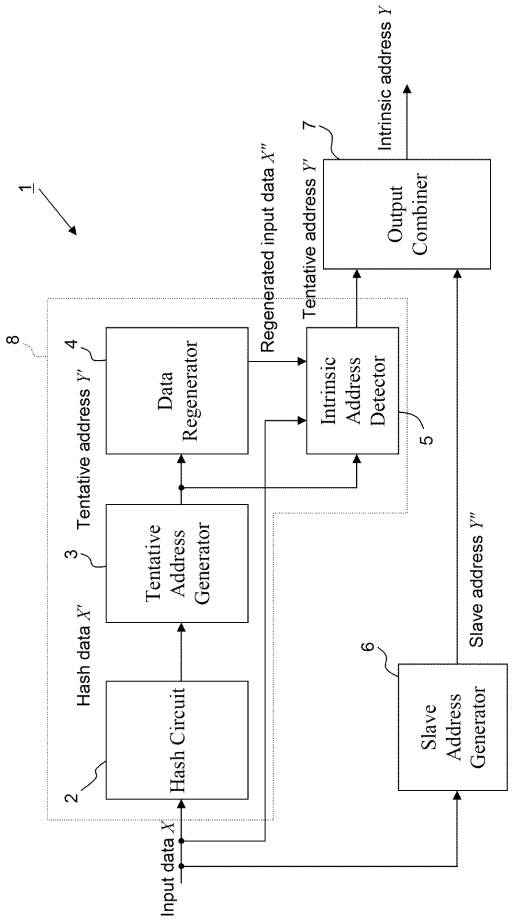
【図２】



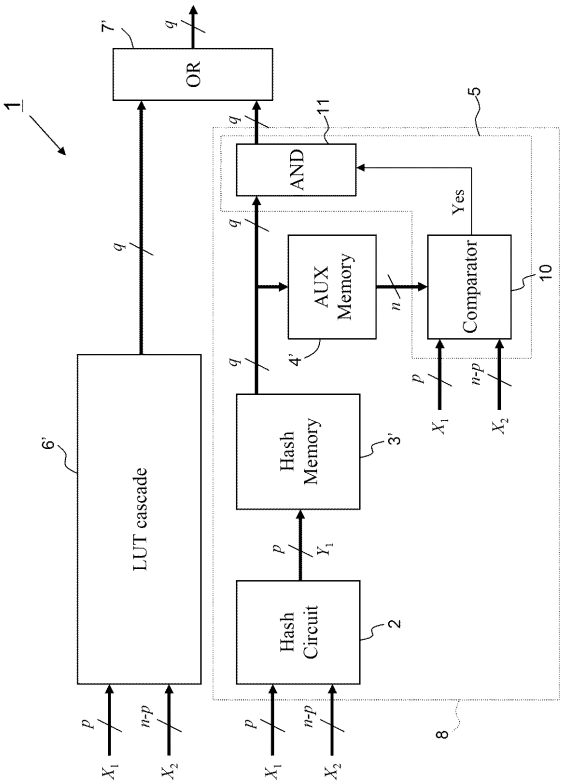
【図３】



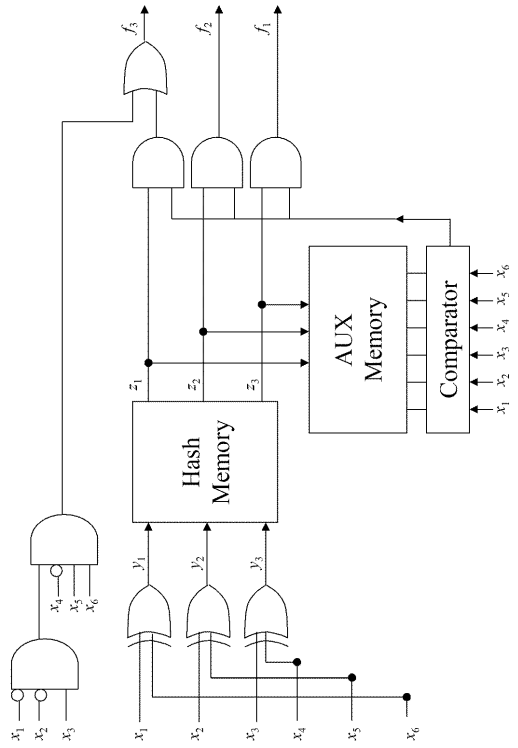
【図４】


1
8
Hash data X'
Tentative address Y'
Regenerated input data X''
Tentative address Y''
Intrinsic address Y
Slave address Y'''
Input data X
2
Hash Circuit
3
Tentative Address Generator
4
Data Regenerator
5
Intrinsic Address Detector
6
Slave Address Generator
7
Output Combiner

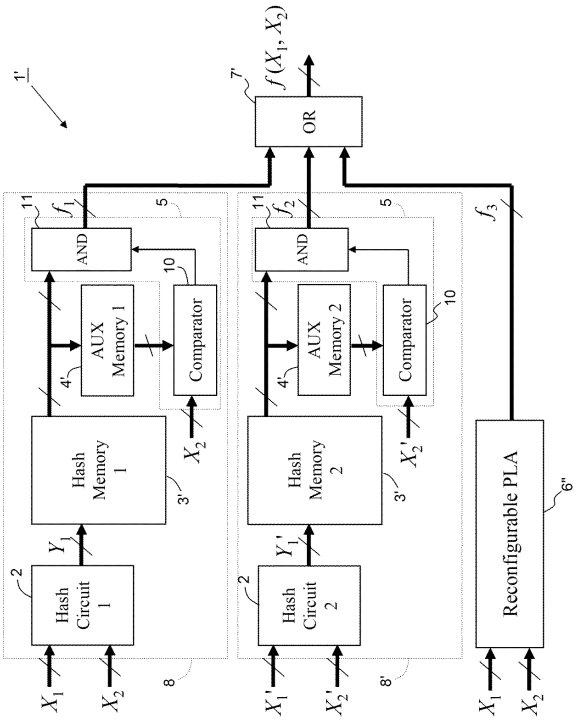
【図５】


1
OR
LUT cascade
Hash Circuit
Hash Memory
AUX Memory
Comparator
AND
Yes
8

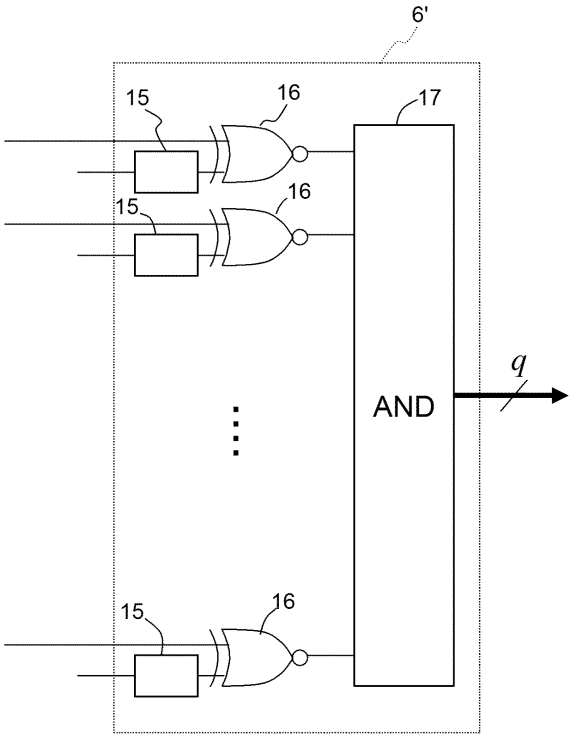
【図６】


Hash Memory
AUX Memory
Comparator

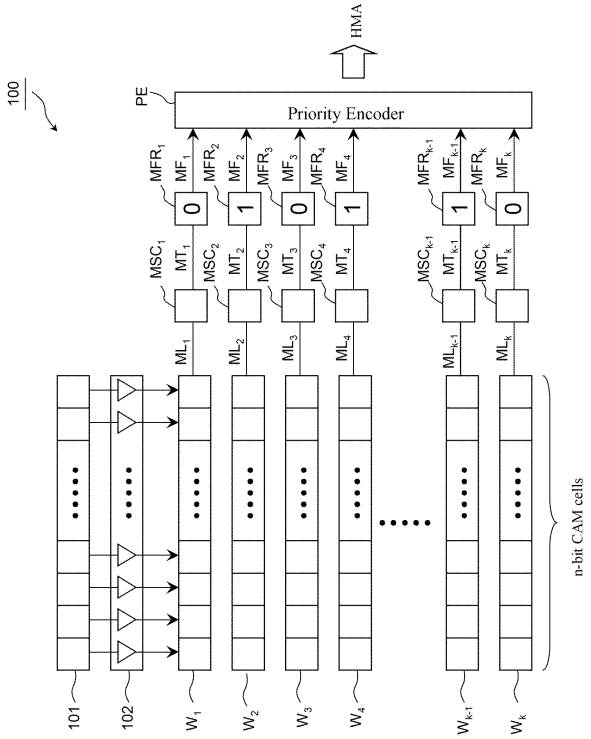
【図７】


1'
OR
AND
AUX Memory 1
AUX Memory 2
Comparator
Hash Memory 1
Hash Memory 2
Hash Circuit 1
Hash Circuit 2
Reconfigurable PLA

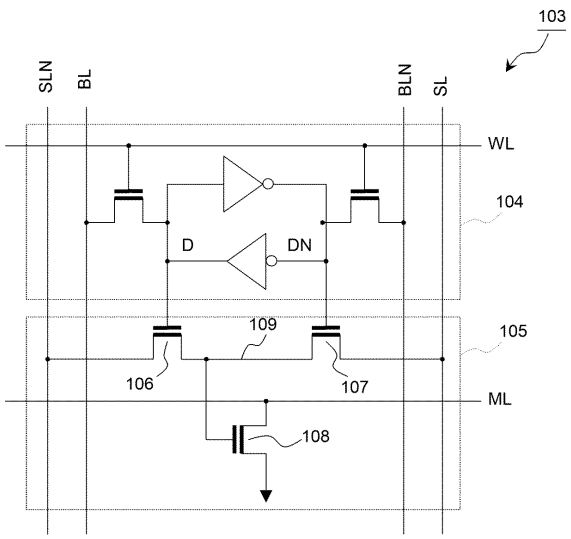
【図８】


6'
15
16
17
AND
q

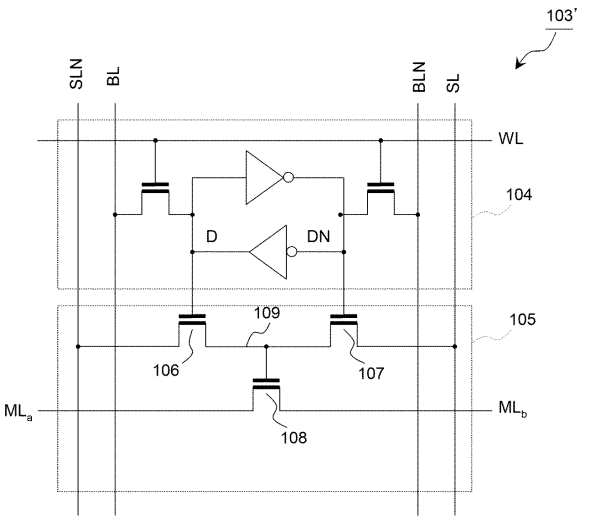
【図９】


HMA
100
PE
Priority Encoder
n-bit CAM cells

【図１０】


103
SLN
BL
BLN
SL
WL
104
D
DN
105
109
106
107
ML
108

【図１１】


103'
SLN
BL
BLN
SL
WL
104
D
DN
105
109
106
107
108

フロントページの続き

(56)参考文献 特開２００４－２２９１６３（ＪＰ，Ａ）
特開昭６４－７６３２０（ＪＰ，Ａ）
Hui Qin,Tsutomu Sasao,Jon.T.Butler, Implementation of LPM address generatorson FPGAs, Reconfigurable Computing: Architectures and Applications. SecondInternational Workshop , ARC 2006. (L, ２００６年 3月 3日, p.170-181, ＵＲＬ, http://www.lsi-cad.com/sasao/Papers/files/ARC2006_qin.pdf

(58)調査した分野(Int.Cl., ＤＢ名)
G11C 15/04
G06F 7/00
G06F 17/30