

(19) 世界知的所有権機関
国際事務局



(43) 国際公開日
2007年10月25日 (25.10.2007)

PCT

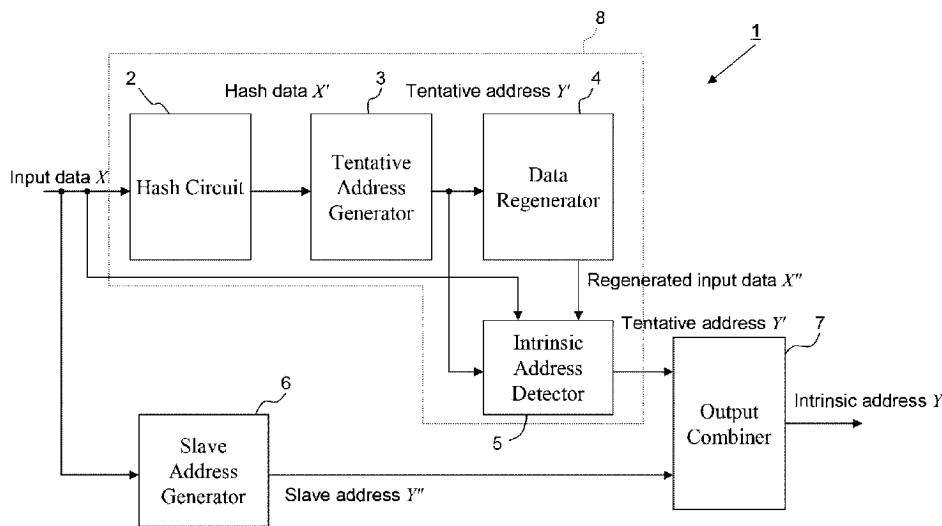
(10) 国際公開番号
WO 2007/119539 A1

- (51) 国際特許分類:
G06F 7/00 (2006.01) G11C 15/04 (2006.01)
- (21) 国際出願番号: PCT/JP2007/056497
- (22) 国際出願日: 2007年3月27日 (27.03.2007)
- (25) 国際出願の言語: 日本語
- (26) 国際公開の言語: 日本語
- (30) 優先権データ:
特願2006-101104 2006年3月31日 (31.03.2006) JP
- (71) 出願人 (米国を除く全ての指定国について): 国立
大学法人九州工業大学 (KYUSHU INSTITUTE OF
TECHNOLOGY) [JP/JP]; 〒8048550 福岡県北九州市
戸畑区仙水町1-1 Fukuoka (JP).
- (72) 発明者; および
- (75) 発明者/出願人 (米国についてのみ): 笹尾 勤 (SASAO,
Tsutomu) [JP/JP]; 〒8208502 福岡県飯塚市川津
680-4 九州工業大学情報工学部内 Fukuoka (JP).
- (74) 代理人: 石田 和人 (ISHIDA, Kazuto); 〒8080135 福岡
県北九州市若松区ひびきの2番1号 北九州学術研
究都市産学連携センターT-302 Fukuoka (JP).
- (81) 指定国 (表示のない限り、全ての種類の国内保護が
可能): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BH,
BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK,
DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT,
HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP,
KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, MG,
MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM,
PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL,
SM, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ,
VC, VN, ZA, ZM, ZW.
- (84) 指定国 (表示のない限り、全ての種類の広域保護が可
能): ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD,
SL, SZ, TZ, UG, ZM, ZW), ユーラシア (AM, AZ, BY,
KG, KZ, MD, RU, TJ, TM), ヨーロッパ (AT, BE, BG,
CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE,

[続葉有]

(54) Title: ADDRESS GENERATOR

(54) 発明の名称: アドレス生成器



(57) **Abstract:** An address generator is provided with features of suppression of power consumption, a simplified device structure, and a small installing area. The address generator is comprised of a hash circuit for outputting (Y_1) that hashes (X_1) with respect to an input vector ($X=(X_1, X_2)$), a tentative address generator for making an address generating function ($f(X)$) a tentative address (A') in the case that a hash collision does not take place with (Y_1) and for making one of an intrinsic address (A) the tentative address (A') in the other cases, a data regenerator for outputting ($X'' = f^{-1}(A')$), an intrinsic address detector for outputting the tentative address (A') in the case that the output (X'') is consistent with the input vector (X) and for outputting an invalid value in the other cases, a complementary address generator for outputting the address generating function ($f(X)$) for the input vector (X) for which the intrinsic address detector outputs the invalid value and for outputting an invalid value in the other cases, and an output combiner for outputting a specific value as an intrinsic address (A) in the case that such a specific value as other than the invalid values exists in the output of the intrinsic address detector and that of the complementary address generator and for outputting an invalid value as the intrinsic address (A) in the other cases.

[続葉有]

WO 2007/119539 A1



IS, IT, LT, LU, LV, MC, MT, NL, PL, PT, RO, SE, SI, SK,
TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW,
ML, MR, NE, SN, TD, TG).

添付公開書類:
— 国際調査報告書

規則4.17に規定する申立て:

— 発明者である旨の申立て (規則 4.17(iv))

2文字コード及び他の略語については、定期発行される各PCTガゼットの巻頭に掲載されている「コードと略語のガイダンスノート」を参照。

(57) 要約: 消費電力を抑え且つデバイスの構造を単純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。入力ベクトル $X = (X_1, X_2)$ に対し X_1 をハッシュ化した Y_1 を出力するハッシュ回路、 Y_1 にハッシュ衝突が生じなければアドレス生成関数 $f(X)$ を仮アドレス A' とし、それ以外は固有アドレス A の一つを A' とする仮アドレス生成器、 $X'' = f^{-1}(A')$ を出力するデータ再生器、 X'' と X とが一致する場合は A' を出力し、それ以外は無効値を出力する固有アドレス検出器、固有アドレス検出器が無効値を出力する X に対し $f(X)$ を出力し、それ以外は無効値を出力する補完アドレス生成器、及び固有アドレス検出器及び補完アドレス生成器の出力のうち無効値以外のものがあればその値を固有アドレス A として出力し、それ以外は無効値を A として出力する出力合成器を備えた。

明 細 書

アドレス生成器

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 $W_1 \sim W_k$ 、 k 個の一致センス回路 $MSC_1 \sim MSC_k$ 、 k 個の一致フラグレジスタ $MFR_1 \sim MFR_k$ 、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 $W_1 \sim W_k$ 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とグラウ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 $W_1 \sim W_k$ 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 $W_1 \sim W_k$ 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 $ML_1 \sim ML_k$ 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 $MSC_1 \sim MSC_k$ に入力される。各一致センス回路 $MSC_1 \sim MSC_k$ 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 $MT_1 \sim MT_k$ に出力

- する。各一致センス出力は、一致フラグレジスタ $MFR_1 \sim MFR_k$ 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 $MF_1 \sim MF_k$ 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 $W_1 \sim W_k$ 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が平行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 $MFR_1 \sim MFR_k$ 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, \dots, 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 Automation 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、 $40000/2^{128}=1.17549 \times 10^{-34}$ 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単純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 $f(X):B^n \rightarrow \{0, 1, \dots, k\}$ ($B=\{0, 1\}$, $k \in \text{自然数}$)において、 k 個の異なる登録ベクトル $a_i \in B^n$ ($i=1, 2, \dots, k$)に対して $f(a_i)=i$ ($i=1, 2, \dots, k$)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($k \ll 2^n$)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 $f_2 f_1 f_0$	登録ベクトル			
	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 $X = (x_1, x_2, \dots, x_n)$ とする。Xの変数の集合を $\{X\}$ で表す。 $\{X_1\} \cup \{X_2\} = \{X\}$ 且つ $\{X_1\} \cap \{X_2\} = \phi$ 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 $f(X) : B^n \rightarrow B^q$ ($B = \{0, 1\}$, $X = (x_1, x_2, \dots, x_n)$, $n, q \in$ 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、 $X_L = X, X_H = \phi$ の場合も考える。

また、関数fの分解表のうちで、 $X_L = (x_1, x_2, \dots, x_{nL})$ 且つ $X_H = (x_{nL+1}, x_{nL+2}, \dots, x_n)$ 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、 $n_L = 3, n_H = 2, \mu = 2$ である。

(例終わり)

[0042] [表3]

表 3: 分解表の例

		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							
X_H	0	0	0								1	0	0	1	1	0	
	0	1	1								1	1	1	1	1	1	
	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 $X = (x_1, x_2, \dots, x_n)$ 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)

$f_1 = x_1 x_2 \vee x_3 x_4 \vee x_5 x_6$ のC尺度は3であるが、 $f_2 = x_2 x_1 x_5 \vee x_2 x_6 \vee x_3 x_4$ 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 $f(x_1, x_2, \dots, x_n)$ 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, \dots, G_s と記す。

[0052] [数2]

$$\begin{aligned}
 R_1 &= G_1(X_1) \\
 R_2 &= G_2(X_2, R_1) \\
 R_3 &= G_3(X_3, R_2) \\
 &\vdots \\
 R_s &= G_s(X_s, R_{s-1}) \\
 f(X) &= R_s \\
 X &= (X_1, X_2, \dots, X_s)
 \end{aligned}$$

各部分関数 G_1, G_2, \dots, 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, \dots, 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の値が大きいつきは、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けて、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]

$$\hat{f}(Y_1, X_2) \tag{1}$$

の分解表(図3)を考える。尚、以下では

[0062] [数8]

$$\hat{f}$$

のように「 $\hat{\quad}$ 」が付された表記をテキスト上では「f $\hat{\quad}$ 」のように記す。Y₁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、2^p>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]

$$\hat{h}(Y_1) = \begin{cases} 0 & \text{(列 } Y_1 \text{ の要素がすべて } 0 \text{ のとき)} \\ \text{列 } Y_1 \text{ での非零要素の最小値} & \text{(列 } Y_1 \text{ に非零要素が存在するとき)} \end{cases} \tag{2a}$$

又は

$$\hat{h}(Y_1) = \max_{b \in B^{n_2}} \hat{f}_1(Y_1, b) \quad (n_2 \text{ は } X_2 \text{ の要素数)} \tag{2b}$$

とし、 $\hat{h}(Y_1)$ をハッシュ・メモリ(Hash memory) 3'で実現する(図5参照)。

[0064] 関数 f^{\wedge} の入力は (Y_1, X_2) であるのに対し、 $h^{\wedge}(Y_1)$ は Y_1 の値のみで f^{\wedge} の値を予測するので、 $h^{\wedge}(Y_1)$ の値は f^{\wedge} とは異なる可能性がある。そこで、 $h^{\wedge}(Y_1)$ 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]

$$q = \lceil \log_2(k + 1) \rceil \quad (3)$$

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^{\wedge} 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、 $f^{\wedge}(Y_1, X_2)$ を

[0067] [数11]

$$\begin{aligned} f^{\wedge}(Y_1, X_2) &= \hat{f}_1(Y_1, X_2) \vee \hat{f}_2(Y_1, X_2), \\ \hat{f}_1(Y_1, X_2) \cdot \hat{f}_2(Y_1, X_2) &= 0 \end{aligned}$$

と関数分解し、関数 $f^{\wedge}(Y_1, X_2)$ 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 $f^{\wedge}_2(Y_1, X_2) = f_2(X_1, X_2)$ 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、 $f^{\wedge}_2(Y_1, X_2)$ 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 $f^{\wedge}_1(Y_1, X_2)$ を主アドレス生成関数(master address generation function)という。また、関数 $f^{\wedge}_2(X_1, X_2)$ 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}) ($i=1, \dots, M$)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 $\hat{f}_{1i}(Y_{1i}, X_{2i})$ を実現する(図7参照)。そして、アドレス生成関数 $\hat{f}(Y_1, X_2)$ を

[0071] [数12]

$$\hat{f}(Y_1, X_2) = \hat{f}_{11}(Y_{11}, X_{21}) \vee \dots \vee \hat{f}_{1k}(Y_{1k}, X_{2k}) \vee \hat{f}_2(Y_1, X_2),$$

$$\hat{f}_{1i}(Y_{1i}, X_{2i}) \cdot \hat{f}_2(Y_1, X_2) = 0 \quad (i = 1, \dots, M)$$

と関数分解し、主アドレス生成関数 $\hat{f}_{1i}(Y_{1i}, X_{2i})$ 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 $\hat{f}_2(Y_1, X_2) = f_2(X_1, X_2)$ 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、 $\hat{f}_2(Y_1, X_2)$ 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 $\hat{f}_{11}(Y_{11}, X_{21})$, $\hat{f}_{12}(Y_{12}, X_{22})$ を実現したとする。主アドレス生成関数 $\hat{f}_{11}(Y_{11}, X_{21})$ 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、 $\hat{f}_{12}(Y_{12}, X_{22})$ 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 $\hat{f}_{11}(Y_{11}, X_{21})$, $\hat{f}_{12}(Y_{12}, X_{22})$ 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を実現される。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 $X' = (Y_{1i}, X_{2i})$ 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、デバイスの構造を単純化し、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を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ることで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単純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 $f(\mathbf{X})$ 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 \mathbf{X} が入力されると、入力ベクトル \mathbf{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]

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

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

[0112] ここで、入力ベクトル \mathbf{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、 $\hat{h}(Y_1)$ 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 $\hat{f}_2(X)$ 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 $\hat{f}_2(X)$ 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 $\hat{h}(Y_1)$ 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 $\hat{f}_2(X)$ 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 Y_1 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 $\hat{f}(Y_1, X_2)$ と記す。関数 $\hat{f}(Y_1, X_2)$ 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 $\hat{h}(Y_1)$ 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、 $A' = \hat{h}(Y_1)$ は正しい値 $A = \hat{f}(Y_1, X_2)$ 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 $\hat{f}(Y_1, X_2)$ 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、 $f_1^{\wedge}(Y_1, X_2)$ 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 $f_1^{\wedge}(Y_1, X_2)$ を式(5)のように関数分解し、 $f_2^{\wedge}(Y_1, X_2)$ 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]

$$\hat{f}(Y_1, X_2) = \hat{f}_1(Y_1, X_2) \vee \hat{f}_2(Y_1, X_2) \quad (5)$$

従って、ハッシュ化された副アドレス生成関数 $f_2^{\wedge}(Y_1, X_2)$ は、主アドレス生成関数 $f_1^{\wedge}(Y_1, X_2)$ が決まれば、一意的に次式(6)により求まる。

[0138] [数18]

$$\hat{f}_2(Y_1, X_2) = \begin{cases} \hat{f}(Y_1, X_2) & (\text{if } \hat{f}_1(Y_1, X_2) = 0), \\ 0 & (\text{otherwise}). \end{cases} \quad (6)$$

[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 $f_1^{\wedge}(Y_1, X_2)$ と副アドレス生成関数 $f_2^{\wedge}(Y_1, X_2)$ 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]

$$f(X_1, X_2) = \hat{f}_1(Y_1, X_2) \vee f_2(X_1, X_2) \quad (7)$$

となる。そこで、OR回路7'は、AND回路11の出力値 $f_1^{\wedge}(Y_1, X_2)$ 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]

$$Y_1 = (y_1, y_2, \dots, y_p), \quad y_i = x_i \oplus x_j \quad (x_i \in \{X_1\}, y_i \in \{Y_1\}, x_j \in \{X_2\}) \quad (8)$$

[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 $X_1 = (x_1, x_2, \dots, x_p)$ 、自由変数を $X_2 = (x_{p+1}, x_{p+2}, \dots, x_n)$ とする。束縛変数 X_1 を $Y_1 = (y_1, y_2, \dots, y_p)$ に置換した関数を $\hat{f}(Y_1, X_2)$ とする。関数 $\hat{f}(Y_1, X_2)$ 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 ($i=1, 2, \dots, p$)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 $x'_{j_i} \in \{X_2\}$ 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 $X = (x_{j_1}, x_{j_2}, \dots, x_{j_p})$ 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 $X_1 = (x_1, x_2, \dots, x_p)$ を

[0149] [数22]

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

で置き換えて得られる関数を $\hat{f}(Y_1, X_2)$ 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]

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

[0151] p次元の2値ベクトルa(∈Y₁)に対してf[^](a, X₂)の非零出力値が2個以上ある場合、f[^](a, X₂)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f₁[^](Y₁, X₂)とする。主アドレス生成関数f₁[^](Y₁, X₂)は次式(11)で表される。

[0152] [数24]

$$f_{1}^{\wedge}(Y_{1}, X_{2}) = \begin{cases} \hat{f}(Y_{1}, X_{2}) & (\text{if } \hat{f}(Y_{1}, X_{2}) = \min_{b \in B^{(n-p)}, \hat{f} \neq 0} \hat{f}(Y_{1}, b)) \\ 0 & (\text{otherwise}) \end{cases} \quad (11)$$

次に、関数f₂[^](Y₁, X₂)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f₁[^](Y₁, X₂)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[^](Y₁)をハッシュ・メモリ3'で実現する。

[0154] [数25]

$$\hat{h}(Y_{1}) = \max_{b \in B^{n-p}} \hat{f}_{1}(Y_{1}, b) \quad (12)$$

[0155] 但し、関数h[^](Y₁)の値は主アドレス生成関数f₁[^](Y₁, X₂)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f₂[^](Y₁, X₂)から次式(13)の置換を行って副アドレス生成関数f₂(Y₁, X₂)を生成する。副アドレス生成関数f₂(Y₁, X₂)は、LUTカスケード6'で実現される。

[0156] [数26]

$$x_{i} = y_{i} \oplus x_{j_{i}} \quad (13)$$

[0157] (例5)

重みがk=7の6変数のアドレス生成関数f(X₁, X₂)の分解表を(表4)に示す。この関数で束縛変数X₁=(x₁, x₂, x₃)を次式(14)のようにY₁に置換し、ハッシュ化した関数f[^](Y₁, X₂)の分解表を(表5)に示す。

[0158] [数27]

$$Y_{1} = (y_{1}, y_{2}, y_{3}) = (x_{1} \oplus x_{6}, x_{2} \oplus x_{5}, x_{3} \oplus x_{4}) \quad (14)$$

[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、(x₁, x₂, x₃)=(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 $f^{\wedge}(X_1, X_2)$ 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 $f_1^{\wedge}(Y_1, X_2)$ 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 $f_2^{\wedge}(Y_1, X_2)$ 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 $h^{\wedge}(Y_1)$ は、 (y_1, y_2, y_3) のみの値を用いて、関数 f_1^{\wedge} の値を予測しており、 f_1^{\wedge} 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_6, x_5, x_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、 $\hat{h}(Y_1)$ 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 \hat{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(15)で表される。ここで、 $p = |Y_1|$ は $\hat{f}(Y_1, X_2)$ の分解表の束縛変数の個数である。

[0172] [数28]

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

(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。 $\alpha = k/2^n$ は、分解表中での非零要素の割合を示す。 $\beta = 1 - \alpha$ は、分解表中での零要素の割合を示す。 $r = n - p = |X_2|$ は、 $\hat{f}(Y_1, X_2)$ 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]

$$P_0 = \beta^{2^r} \quad (16)$$

$$P_1 = 1 - \beta^{2^r} \quad (17)$$

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]

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

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]

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

(証明終わり)

[0177] 例えば、 $k/2^p=1/4$ とすると、 $\delta \simeq 0.8854$ となる。また、 $1/4 \leq k/2^p \leq 1/2$ の領域では、 δ は、 $k/2^p$ 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、 $2^p \times q = 2^{13} \times 11$ bits。従って、総メモリ量は、 $2^{13} \times 11 \times 15 = 1,351,680$ bits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]

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

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 $1730 \times (1 - 0.901) = 170$ 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 $2^{10} \times 8$ bits。最終段のセルのメモリ量は、高々 $2^{10} \times 11$ bits。カスケードの総メモリ量は、 $2^{10} \times 8 \times 15 + 2^{10} \times 11 =$

134, 144bits。ハッシュ・メモリ3'の容量は、 $2^{13} \times 11 = 901112$ bits。補助メモリ4'の容量は、 $2^{11} \times 40 = 81920$ bits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]

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

束縛変数の個数は、 $p=q+2=13$ 。分解表の列の個数は $2^p=2^{13}=8192$ 。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_{11}^{\wedge} 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 $f-f_{11}^{\wedge}$ 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}^{\wedge} 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 $f_{21}^{\wedge}(X_1, X_2)$ 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PLA 6")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]

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

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

カ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]

$$\left(\left\lfloor \frac{n}{2} \right\rfloor + \left\lfloor \frac{2n-1}{3} \right\rfloor \right) k + \left\lfloor \frac{\frac{k}{2}-1}{3} \right\rfloor q \quad (23)$$

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]

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

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]

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

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)

請求の範囲

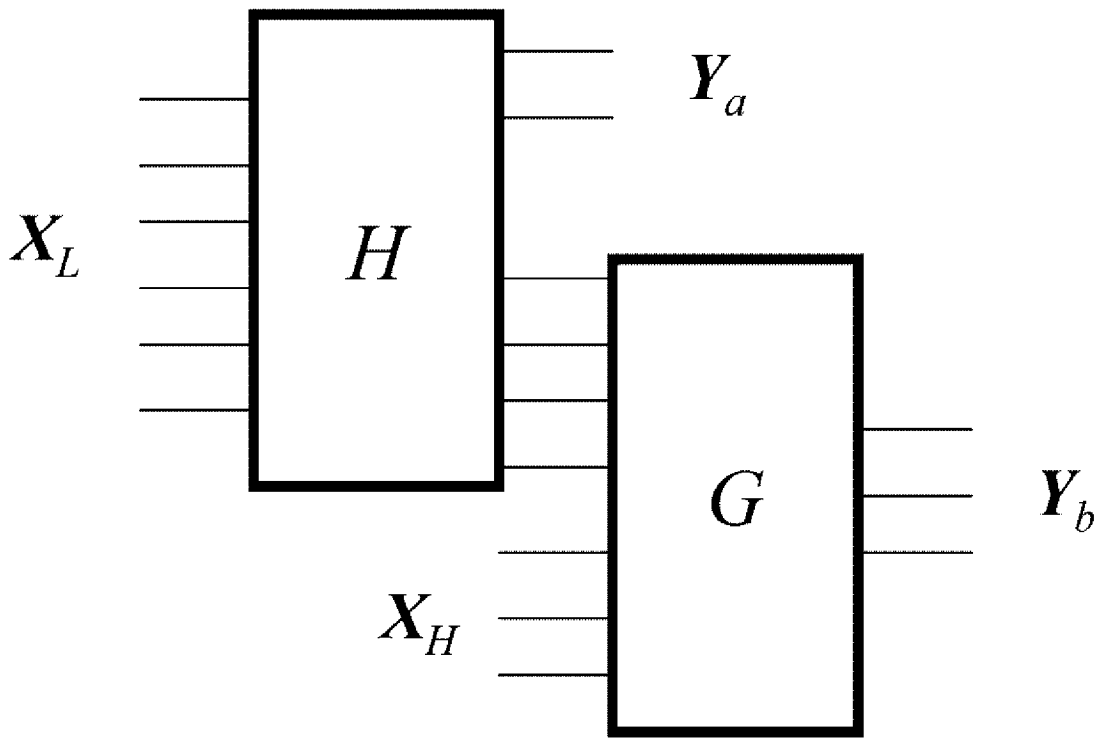
- [1] 入力される2値ベクトル(以下「入力ベクトル」という。)Xに対し、当該入力ベクトルXが登録ベクトルの場合にはそれに対応する固有アドレスAを出力し、それ以外の場合には無効値を出力するアドレス生成関数 $f(X)$ の演算を行うアドレス生成器であつて、
- A. 1乃至複数個設けられ、前記入力ベクトルXが前記登録ベクトル集合の所定の部分集合に属す場合にはそれに対応する固有アドレスAを出力し、それ以外の場合には無効値を出力する主アドレス生成器と、
- B. すべての前記主アドレス生成器の出力が無効値となり且つ前記登録ベクトル集合に属す前記入力ベクトルXに対しては、対応する固有アドレスAを出力し、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つて、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、

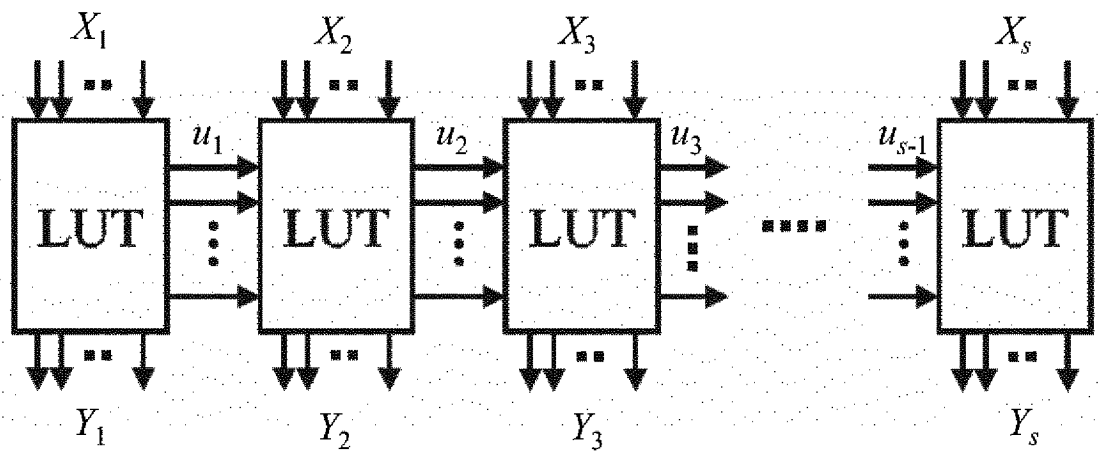
を備えたことを特徴とするアドレス生成器。

- [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_{ii} 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。
- [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。
- [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。
- [5] 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請求項1記載のアドレス生成器。

[図1]



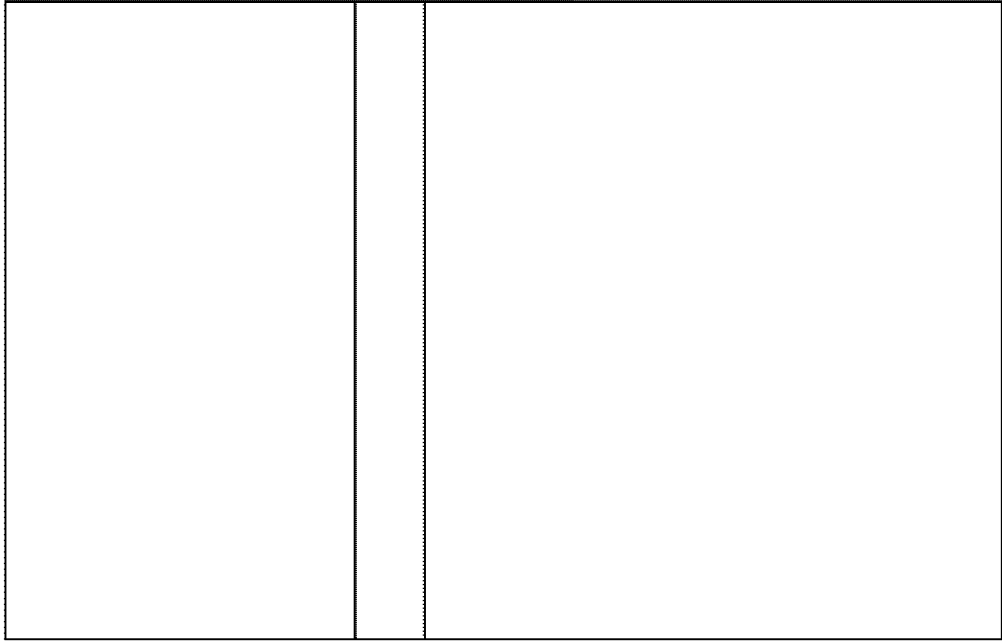
[図2]



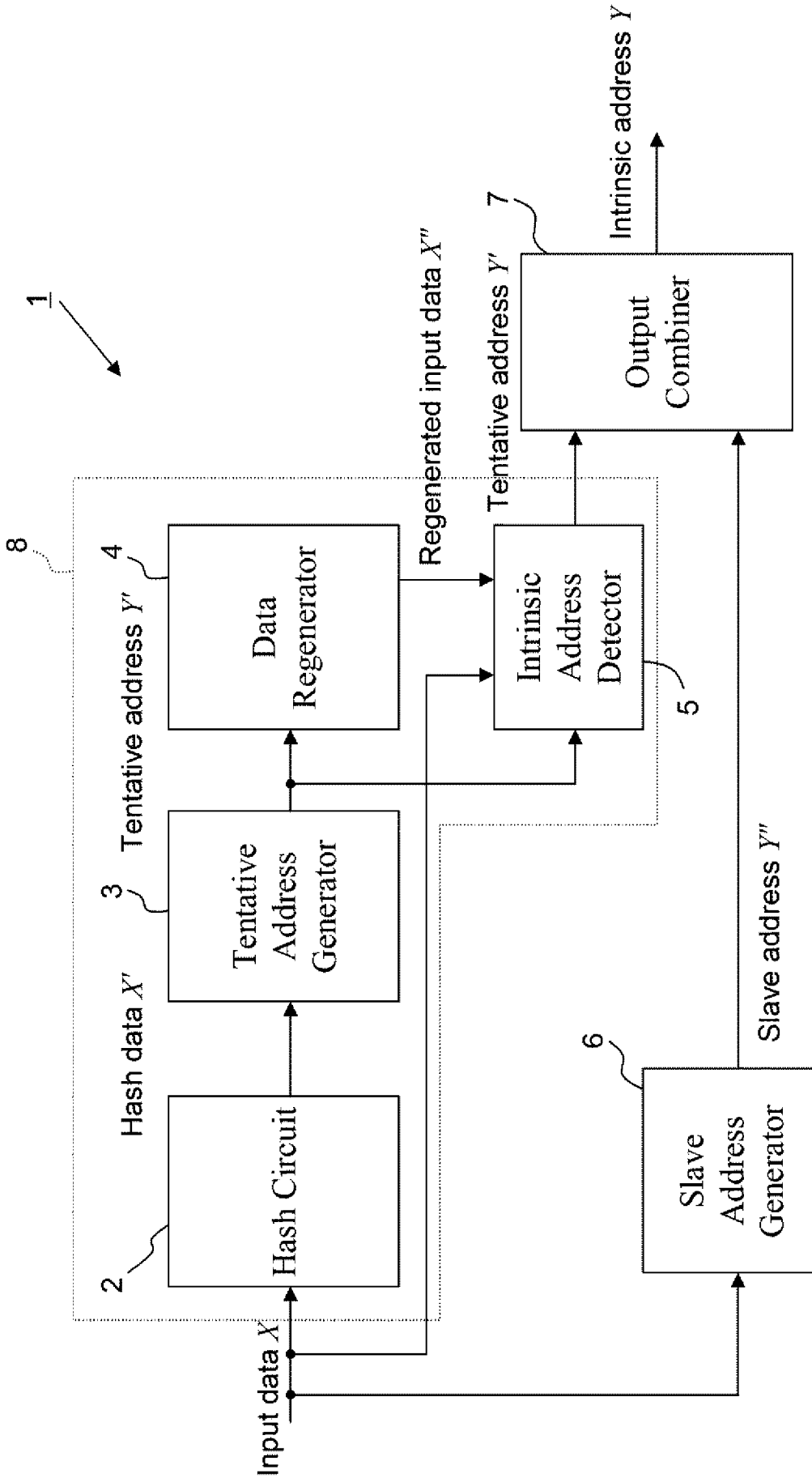
[\otimes 3]

Y_1

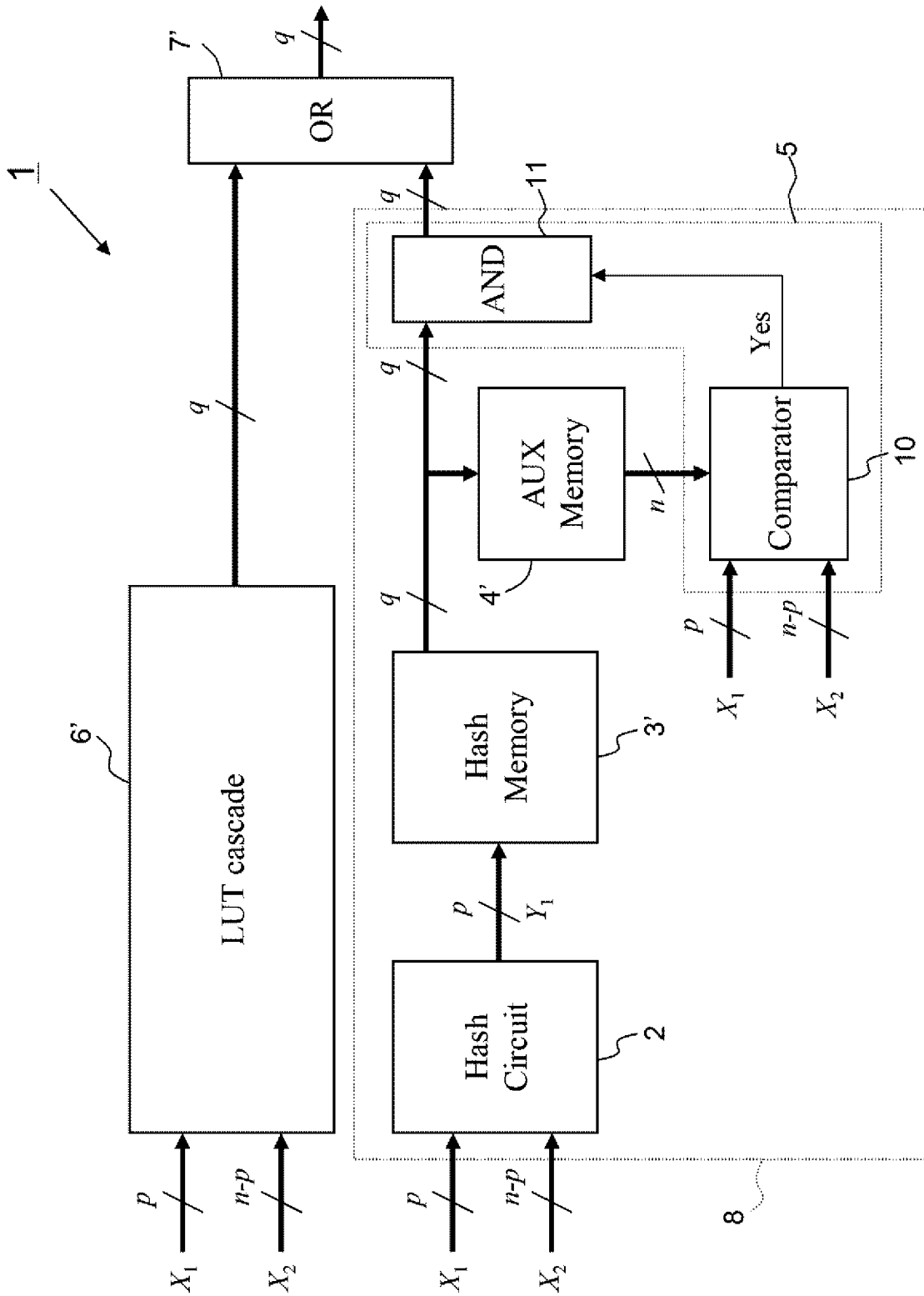
X_2



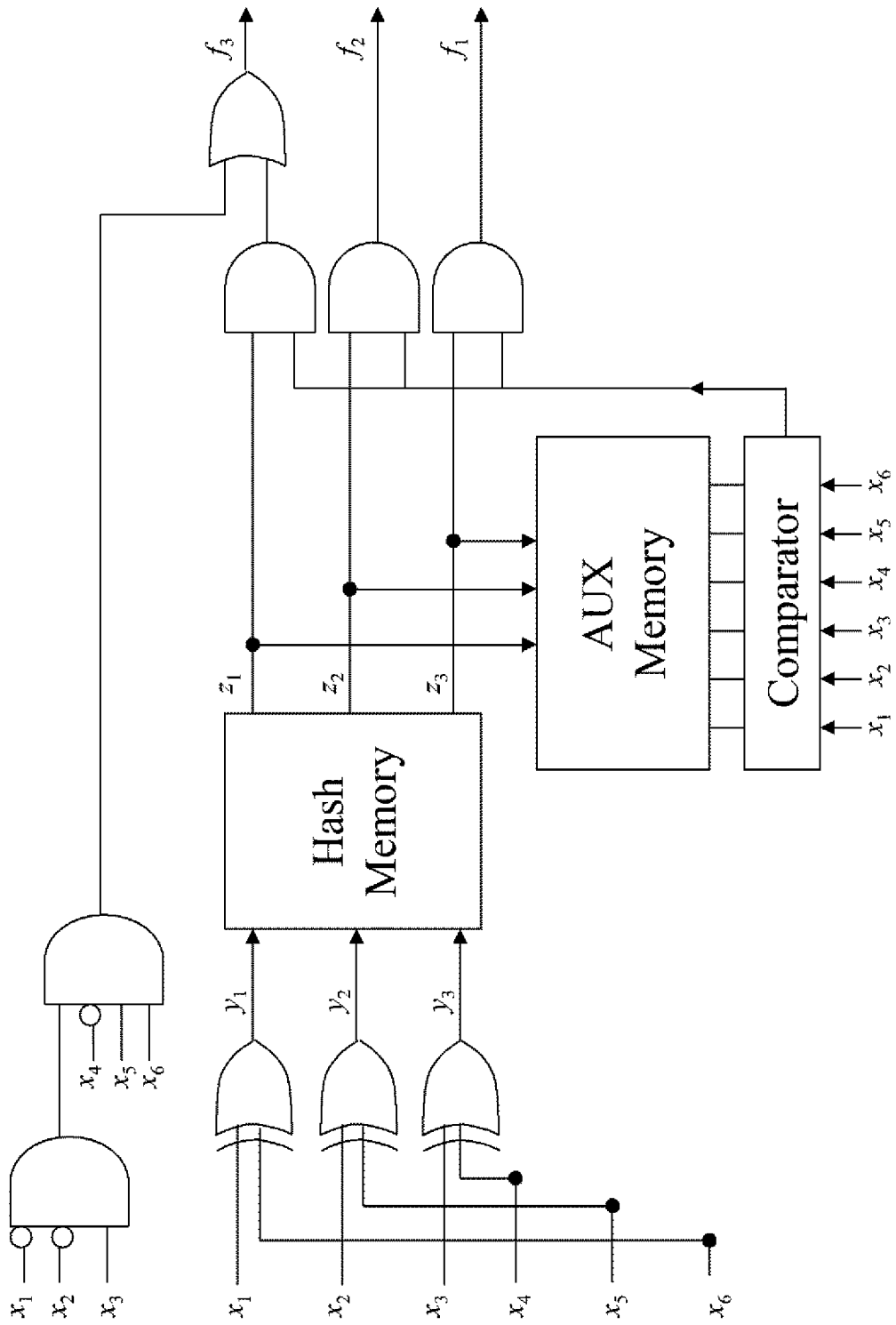
[図4]



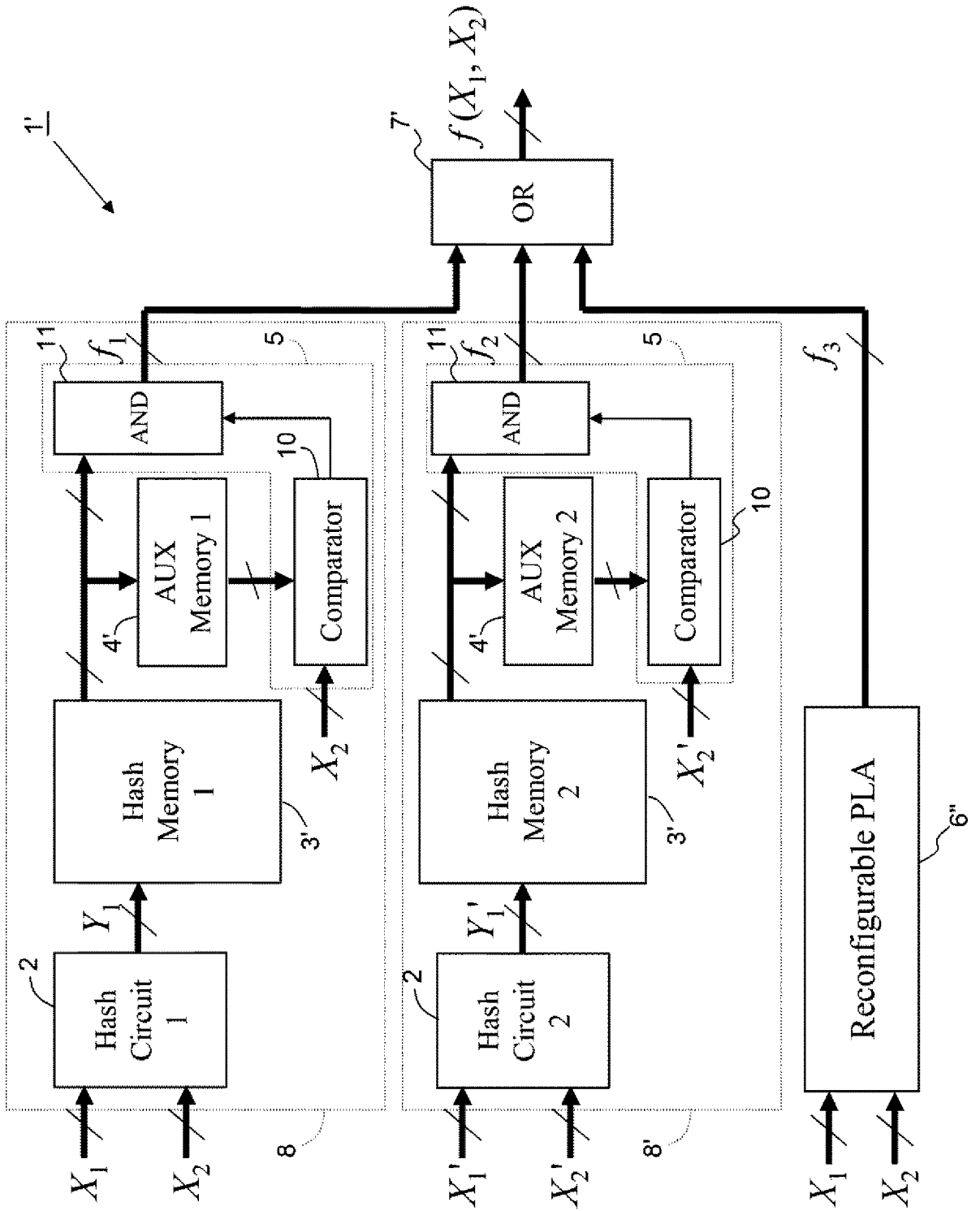
[5]



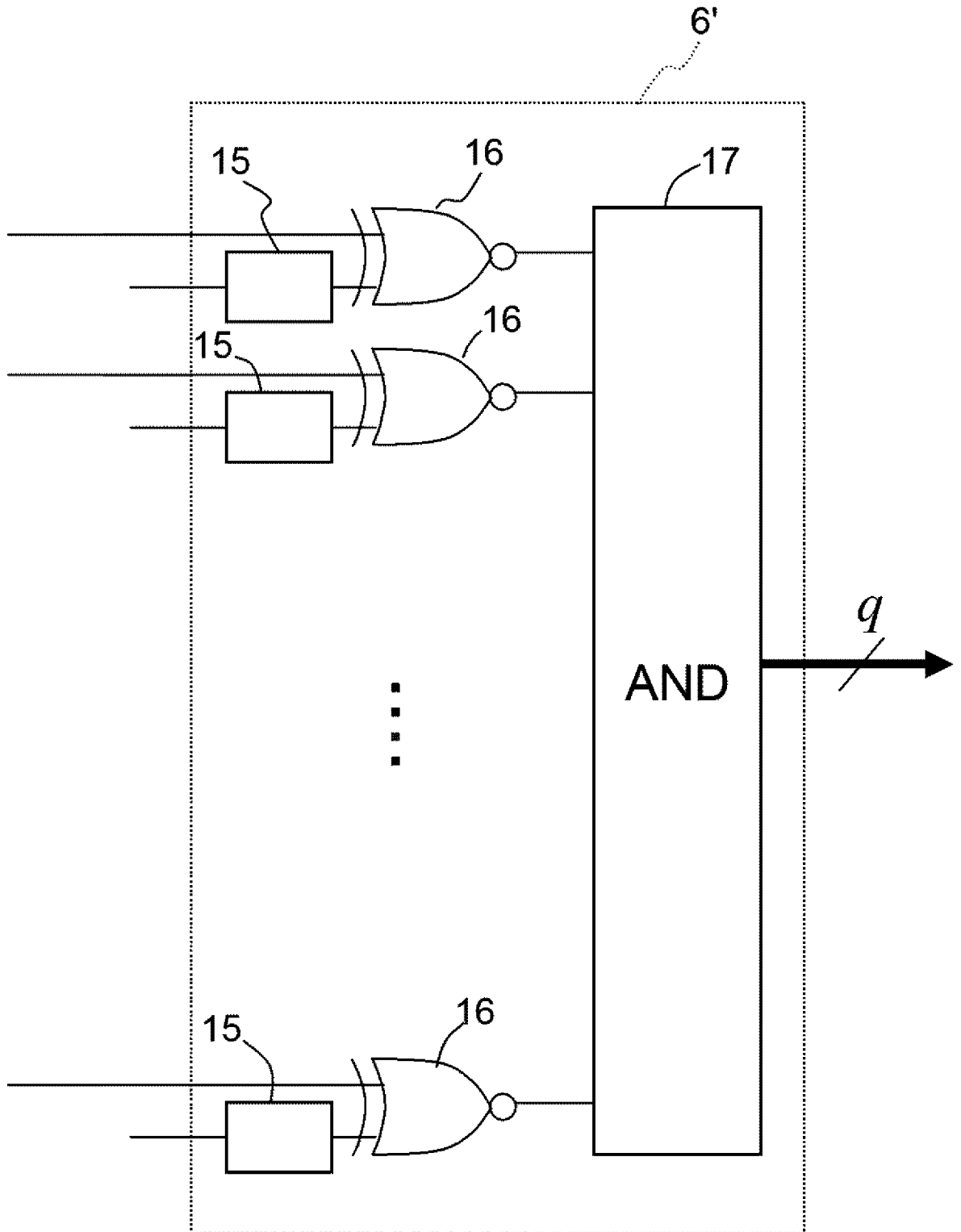
[図6]



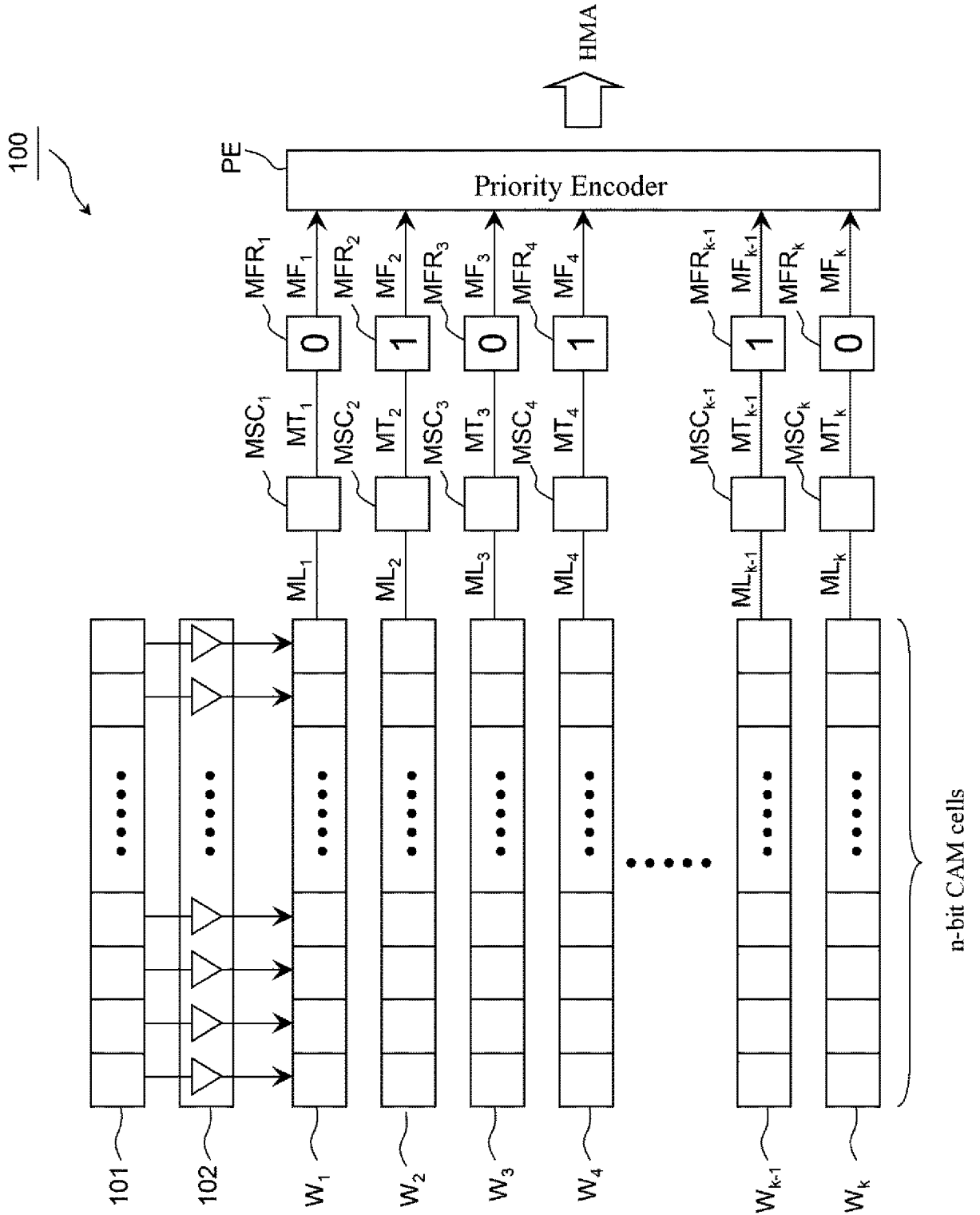
[図7]



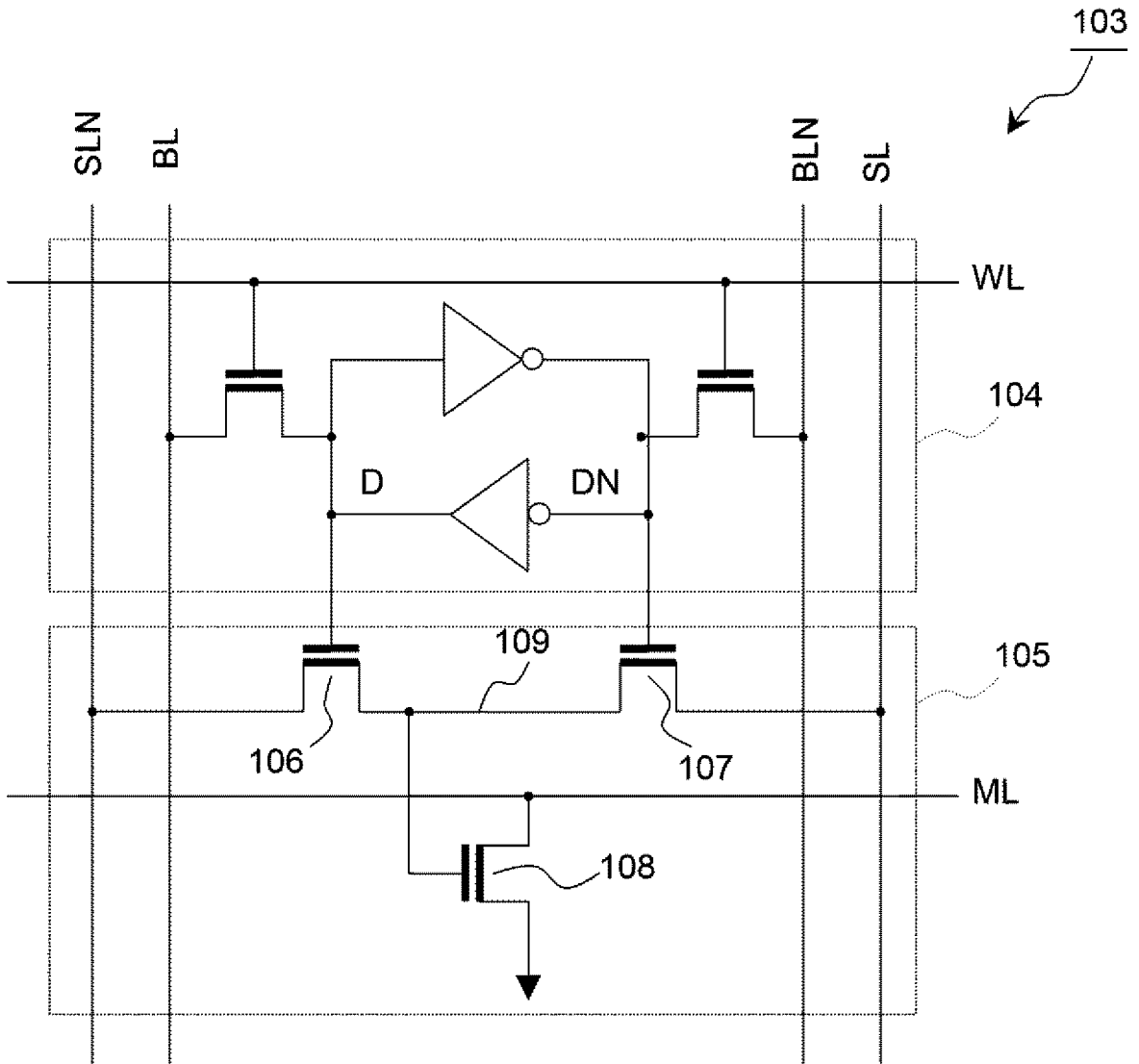
[図8]



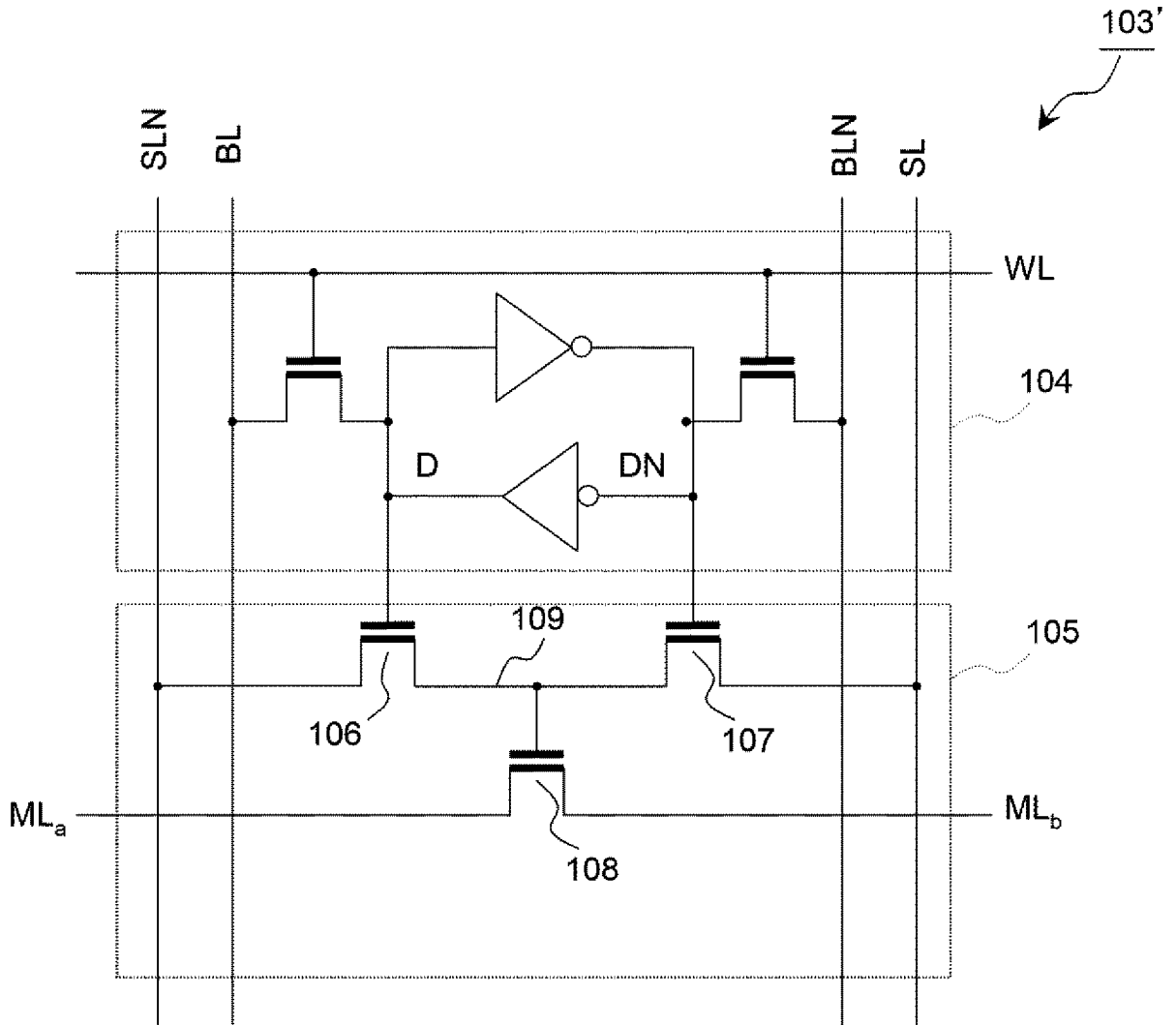
[9]



[10]



[11]



INTERNATIONAL SEARCH REPORT

International application No.

PCT/JP2007/056497

A. CLASSIFICATION OF SUBJECT MATTER
G06F7/00(2006.01) i, G11C15/04(2006.01) i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F7/00, G11C15/04

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
 Jitsuyo Shinan Koho 1922-1996 Jitsuyo Shinan Toroku Koho 1996-2007
 Kokai Jitsuyo Shinan Koho 1971-2007 Toroku Jitsuyo Shinan Koho 1994-2007

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y A	JP 2004-229163 A (International Business Machines Corp.), 12 August, 2004 (12.08.04), Full text; Fig. 2 & US 2004/0205056 A1	1-4 5
Y A	JP 64-076320 A (Fujitsu Ltd.), 22 March, 1989 (22.03.89), Page 3, upper left column, line 2 to upper right column, line 14 (Family: none)	1-4 5

Further documents are listed in the continuation of Box C. See patent family annex.

* Special categories of cited documents:	"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A" document defining the general state of the art which is not considered to be of particular relevance	"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E" earlier application or patent but published on or after the international filing date	"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&" document member of the same patent family
"O" document referring to an oral disclosure, use, exhibition or other means	
"P" document published prior to the international filing date but later than the priority date claimed	

Date of the actual completion of the international search
31 May, 2007 (31.05.07)

Date of mailing of the international search report
12 June, 2007 (12.06.07)

Name and mailing address of the ISA/
Japanese Patent Office

Authorized officer

Facsimile No.

Telephone No.

INTERNATIONAL SEARCH REPORT

International application No.

PCT/JP2007/056497

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	Hui Qin, Tsutomu Sasao, Jon.T.Butler, Implementation of LPM address generators on FPGAs, Reconfigurable Computing: Architectures and Applications. Second International Workshop, ARC2006. (Lecture Notes in Computer Science Vol. 3985), 2006.03.03, p.170-81	5

A. 発明の属する分野の分類 (国際特許分類 (IPC)) Int.Cl. G06F7/00(2006.01)i, G11C15/04(2006.01)i		
B. 調査を行った分野 調査を行った最小限資料 (国際特許分類 (IPC)) Int.Cl. G06F7/00, G11C15/04		
最小限資料以外の資料で調査を行った分野に含まれるもの 日本国実用新案公報 1922-1996年 日本国公開実用新案公報 1971-2007年 日本国実用新案登録公報 1996-2007年 日本国登録実用新案公報 1994-2007年		
国際調査で使用した電子データベース (データベースの名称、調査に使用した用語)		
C. 関連すると認められる文献		
引用文献の カテゴリー*	引用文献名 及び一部の箇所が関連するときは、その関連する箇所の表示	関連する 請求の範囲の番号
Y A	JP 2004-229163 A (インターナショナル・ビジネス・マシーン・コーポレーション) 2004.08.12, 全文, 第2図 & US 2004/0205056 A1	1-4 5
Y A	JP 64-076320 A (富士通株式会社) 1989.03.22, 第3頁左上欄第2行-右上欄第14行 (ファミリーなし)	1-4 5
<input checked="" type="checkbox"/> C欄の続きにも文献が列挙されている。 <input type="checkbox"/> パテントファミリーに関する別紙を参照。		
* 引用文献のカテゴリー 「A」特に関連のある文献ではなく、一般的技術水準を示すもの 「E」国際出願日前の出願または特許であるが、国際出願日以後に公表されたもの 「L」優先権主張に疑義を提起する文献又は他の文献の発行日若しくは他の特別な理由を確立するために引用する文献 (理由を付す) 「O」口頭による開示、使用、展示等に言及する文献 「P」国際出願日前で、かつ優先権の主張の基礎となる出願日の後に公表された文献 「T」国際出願日又は優先日後に公表された文献であって出願と矛盾するものではなく、発明の原理又は理論の理解のために引用するもの 「X」特に関連のある文献であって、当該文献のみで発明の新規性又は進歩性がないと考えられるもの 「Y」特に関連のある文献であって、当該文献と他の1以上の文献との、当業者にとって自明である組合せによって進歩性がないと考えられるもの 「&」同一パテントファミリー文献		
国際調査を完了した日 31.05.2007	国際調査報告の発送日 12.06.2007	
国際調査機関の名称及びあて先 日本国特許庁 (ISA/J P) 郵便番号100-8915 東京都千代田区霞が関三丁目4番3号	特許庁審査官 (権限のある職員) 緑川 隆 電話番号 03-3581-1101 内線 3521	5E 2950

C (続き) . 関連すると認められる文献		
引用文献の カテゴリー*	引用文献名 及び一部の箇所が関連するときは、その関連する箇所の表示	関連する 請求の範囲の番号
A	Hui Qin, Tsutomu Sasao, Jon.T.Butler, Implementation of LPM address generators on FPGAs, Reconfigurable Computing: Architectures and Applications. Second International Workshop, ARC2006. (Lecture Notes in Computer Science Vol. 3985), 2006.03.03, p.170-81	5