クラスタリングによる書き込みビット数削減と誤り訂正を実現する不揮発メモリを対象とした符号の構成手法

古城 辰朗 † 多和田 雅師 † 柳澤 政生 † 戸川 望 †

† 早稲田大学大学院基幹理工学研究科情報理工・情報通信専攻

デバイスの微細化によって不揮発メモリに保存されている値が破壊されるリスクが増大する．メモリの値を破壊から守る手法として誤り訂正符号を用いてメモリを構成することが挙げられる．誤り訂正符号を用いて構成したメモリでは符号語を書き込む際に反転するビット数が多いため，書き込みエネルギーが大きくなるという欠点があり，加えて，不揮発メモリの書き込みエネルギーは通常のメモリの 10 倍以上大きい．そのため，誤り訂正符号を用いて不揮発メモリを構成した場合，書き込むビット数を削減することが強く要求される．本稿では，誤り訂正符号の符号語をクラスタリングし，各クラスタに値を割り当てることで，書き込みビット数削減と誤り訂正を実現する符号を構成する．このような符号を構成するために効果的なクラスタリング手法を提案する．実験結果より，提案アルゴリズムで生成した符号を利用してメモリを構成したとき，アプリケーションに対して書き込みビット数を最大 28.2% 削減した．

A Clustering-based Bit-Write-Reducing and Error-Correcting Generation Method Targeting Non-Volatile Memories

Tatsuro KOJO† Masashi TAWADA† Masao YANAGISAWA† Nozomu TOGAWA†

† Dept. of Computer Science and Communications Engineering, Waseda University.

In this paper, we propose a clustering-based bit-write-reducing and error-correcting code generation method targeting non-volatile memories. The method is based on clustering error-correcting codewords and generates a new code which reduces bit flipping between codewords but still has a same error-correcting ability as the original ones. Experimental results demonstrate the efficiency of the proposed method.

1 はじめに

近年，モバイル機器の普及に伴い，消費電力の小さなメモリや電源を切っても記憶内容を保持し続ける不揮発性のメモリの需要が高まっている．不揮発メモリは高速な読み出し，少ないリーク電力，高集積が可能であるといった利点がある一方，デバイスの微細化によるクロストークや放射線の影響によってメモリに保存されている値が破壊されやすい欠点がある [10].

メモリの値を破壊から守る手法として誤り訂正符号を用いてメモリを構成することが挙げられる．誤り訂正符号を用いてメモリを構成した場合には，保存したい値を符号語にエンコードして，その符号語をメモリに書き込むことで値を保存する．代表的な誤り訂正符号として，ハミング符号，BCH 符号，リードソロモン符号などがある．

メモリに符号語を書き込むとき，符号語の各ビットについて書き込むビットが既に書き込まれているビットと同じであればそのビットの書き込みは行わないものとする．誤り訂正符号を用いて不揮発メモリを構成した場合，誤り訂正符号は符号語同士のハミング距離が大きいため，メモリに符号語を書き込むときの書き込みビット数が増大する．メモリに書き込むビット数が多いほど，消費エネルギーも増えるため，誤り訂正符号を用いた不揮発メモリでは符号語の書き込みにかかるエネルギーが増大するという欠点がある．加えて，不揮発メモリの書き込みエネルギーは通常のメモリの 10 倍以上大きい [3]．そのため，誤り訂正符号を用いて不揮発メモリを構成した場合，書き込むビット数を削減することが強く要求される．

不揮発メモリを対象とした書き込みビット数を削減する研究として [4, 7, 12, 13] がある．Zhou らの手法では保存したい値を表すビット列をメモリに書き込むとき，書き込むビットが書き込まれているビットと同じであればそのビットを書き込まないことで書き込みビット数を削減する [13]．本稿では手法 [13] を前提とする．Cho らの手法ではメモリに保存する値に 1 ビットの冗長部分を付加する [4]．付加ビットが 1 であれば値をビット反転し，付加ビットが 0 であれば値はビット反転しない．書き込むビット長の半数以上に書き込む場合に値をビット反転することで書き込みビット数を削減する．多和田らの手法では誤り訂正符号を元に符号語間の最大ハミング距離を制限する符号を生成することで書き込みビット数を削減する [12]．古城らの手法では最大ハミング距離と最小ハミング距離を制約する符号を生成することで，最悪の場合の書き込みビット数が削減できる誤り訂正符号を生成する [7].

本稿ではまず，t ビットの誤り訂正能力を持つ誤り訂正符号を用意する．その符号語をクラスタリングし，クラスタをノードとするグラフを生成する．各クラスタに値を割り当てることで，クラスタに属する符号語はクラスタに割り当てた値を表すことにする．このようにして値と符号語が 1 対多に対応した符号が生成される．生成したグラフのエッジが条件を満足するとき，符号は t ビットの誤り訂正能力を維持しつつ書き込みビット数を制限できることを示す．符号を生成するために効果的なクラスタリング手法を提案する．提案手法を用いて符号を生成すると，符号長が n ビットのとき，メモリに符号語を書き込むときの最大書き込みビット数は $\lfloor n/2 \rfloor$ に制約でき，平均書き込みビット数を削減できる．

提案手法で生成した符号を用いてメモリを構成し，アプリケーションを動作させたとき，ハミング符号を用いて構成したメモリに対して書き込みビット数を最大 25.9%，BCH 符号 + 反復符号を用いて構成したメモリに対して書き込みビット数を最大 28.2% 削減した．

2 誤り訂正符号

値に符号語が割り当てられているとき，符号語の集合を符号と呼ぶ．符号 V に属するある符号語を v_x, v_y と表す．v_x と v_y のハミング距離を $H(v_x, v_y)$ と表す．v_x のハミング重みを $W_H(v_x)$ と表す．

最小 (最大) ハミング距離とは，符号語間のハミング距離の最小値 (最大値) である．誤り訂正符号では，符号の最小ハミング距離を制約することにより誤り訂正が可能となる．一般に，最小ハミング距離が $2t+1$ のとき，t ビットの誤り訂正能力を持つ．

誤り訂正符号にはハミング符号が知られている．ハミング符号は，符号語間の最小ハミング距離が 3 であり 1 ビットの誤り訂正が可能な完全符号である [6]．自然数 m を用いると，

符号長 [ビット] : $n = 2^m - 1$
情報量 [ビット] : $k_p = n - m$

が成立する．これを $(n, k_p, 3)$ ハミング符号と表す．

ハミング符号や BCH 符号などの誤り訂正符号は最小ハミング距離が制約されていることによって誤り訂正能力は有するが，最大ハミング距離は制約されていない．例えば，(7, 4, 3) ハミング符号の最大ハミング距離は 7 であり，符号長に等しい．つまり，ハミング符号では最悪の場合，メモリに書き込まれた符号長分の書き込みビット数が発生する可能性がある．(7, 4, 3) ハミング符号を用いて構成したメモリでは，1 回の書き込みでの平均書き込みビット数は 3.5 ビットである．本稿では，メモリに誤り訂正符号を用いて構成した場合に書き込みビット数を削減することが目標となる．

3 メモリアーキテクチャ

誤り訂正符号を用いて構成した不揮発メモリを考える．メモリに値を保存するときにはエンコーダを用いて値を符号語にエンコードし，メモリから値を取り出すときにはデコーダを用いて符号語を値にデコードする．値を符号化して保存するメモリのメモリアーキテクチャには 2 つの種類がある．メモリに書き込まれている符号語を考慮せずエンコードする方式（コンテクスト独立）と考慮してエンコードする方式（コンテクスト依存）である．図 1(a) にコンテクスト独立符号化のメモリアーキテクチャを，図 1(b) にコンテクスト依存符号化のメモリアーキテクチャをそれぞれ示す [9].

本稿では，図 1(b) に示すメモリアーキテクチャを用いる．メモリに値を保存するとき，まずメモリに書き込まれている符号語を読み出す．読み出した符号語を基に値を符号語にエンコードする．メモリから値を取り出すときには，メモリに書き込まれている符号語を読み出し，読み出した符号語をデコードすることで元の値を得ることができる．このメモリアーキテクチャに対して，値と符号語が 1 対多に対応した符号を用いたメモリを構成することができる．

4 符号語のクラスタリング

本章では，いくつかの条件を満たすように誤り訂正符号の符号語をクラスタリングして，各クラスタに値を割り当てることで，書き込みビット数削減と誤り訂正を実現する符号が生成できることを証明する．

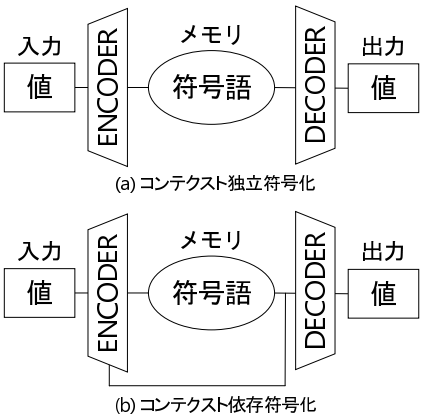

図 1 エンコーダ/デコーダを用いた不揮発メモリのメモリアーキテクチャ．

t ビットの誤り訂正能力を持つ誤り訂正符号を考える．誤り訂正符号の情報量を k_p とするとき，符号語を $v_0, \ldots, v_{2^{k_p}-1}$ とする．符号語 v_i の符号長は n ビットである．すなわち，k_p ビットの値をエンコードすることで n ビットの符号語を得る．符号語同士のハミング距離は $2t+1$ 以上であり，誤り訂正能力は t ビットである．この誤り訂正符号を $(n, k_p, 2t+1)$ 誤り訂正符号と表す．

定数 $S \geq 2t+1$ を与えたとき，$(n, k_p, 2t+1)$ 誤り訂正符号から以下のようなグラフを生成できる．

1. 誤り訂正符号の符号語 $v_0, \ldots, v_{2^{k_p}-1}$ をノードとする．
2. 符号語 v_i, v_j のハミング距離が S 以下であれば，エッジ $e_{ij} = (v_i, v_j)$ を結ぶ．

ノードの集合を $V = \{v_0, \ldots, v_{2^{k_p}-1}\}$，エッジの集合を E とするとき，このグラフを S-バウンドグラフ $G = (V, E)$ と呼ぶ．図 2(a) に S-バウンドグラフの例を示す．詳細は例 1 で述べる．

r を冗長量とし，$k_r = k_p - r$ とする．冗長量とは，1 つのクラスタに属するノード数を定める定数である．ノードの集合 V から以下の条件を満たすクラスタ $c_i (i = 0, \ldots, 2^{k_r} - 1)$ を生成する．

(C1) $c_0 \cup \cdots \cup c_{2^{k_r}-1} = V$
(C2) $c_i \cap c_j = \emptyset$ $(0 \leq (i, j) \leq 2^{k_r} - 1, i \neq j)$
(C3) $|c_i| = 2^r$ $(0 \leq i \leq 2^{k_r} - 1)$

以上の条件 (C1)–(C3) をクラスタリング条件と呼ぶ．$C = \{c_0, \ldots, c_{2^{k_r}-1}\}$ をクラスタの集合とする．クラスタ c_i, c_j $(i \neq j)$ をクラスタの集合 C に属するクラスタとする．c_i, c_j が以下の条件 (F1), (F2) を満たすとき，c_i, c_j をエッジ $e'_{ij} = (c_i, c_j)$ で結ぶ．

(F1) クラスタ c_i に属する任意のノード v_k から，クラスタ c_j に属するあるノード v_l に対してエッジ e_{kl} が存在する．
(F2) クラスタ c_j に属する任意のノード v_l から，クラスタ c_i に属するあるノード v_k に対してエッジ e_{lk} が存在する．

以上の条件 (F1), (F2) を S ビット反転条件と呼ぶ．S-バウンドグラフをもとに，クラスタリング条件からクラスタの集合 C を生成し，S ビット反転条件からクラスタ間のエッジの集合 E' を生成する．クラスタの集合 C とクラスタ間のエッジの集合 E' からクラスタグラフ $G' = (C, E')$ を得る．図 2(b) にクラスタグラフの例を示す．詳細は例 1 で述べる．

$D = \{d_0, \ldots, d_{2^{k_r}-1}\}$ を k_r ビットの値の集合とする．各値は $d_0 = 00\cdots00, d_1 = 00\cdots01, \ldots, d_{2^{k_r}-1} = 11\cdots11$ となる．値 $d_i \in D$ をクラスタ $c_i \in C$ に割り当てる．つまり，クラスタ $c_i \in C$ に属する全ての符号語は値 $d_i \in D$ を表すことにする．

定理 1. 値 $d_i, d_j (i \neq j)$ をクラスタ c_i, c_j にそれぞれ割り当てる．クラスタ c_i に属する任意のノードを v_k とする．クラスタグラフ $G' = (C, E')$ が完全グラフとなるとき，クラスタ c_j に属するあるノード v_l に対して，ハミング距離 $H(v_k, v_l) \leq S$ が成立する．

証明 1. クラスタ c_i, c_j 間には S-ビット反転条件を満足するエッジが必ず存在するため，上記定理が成立する． □

系 1. 値 d_i はエンコードして v_k に，値 d_j はエンコードして v_l になるとする．クラスタグラフ $G' = (C, E')$ が完全グラフとなるとき，値 d_i を表す符号語 v_k から値 d_j を表す符号語 v_l へ書き込みビット数 S ビット以下で遷移できる．

定理 2. クラスタ c_i に属する符号語 v_k は t ビットの誤り訂正能力を持つ．

証明 2. 符号語 v_k は $(n, k_p, 2t+1)$ 誤り訂正符号の符号語であるため，上記定理が成立する． □

以上の定理より，クラスタグラフが完全グラフとなるような符号を用いたメモリでは，クラスタ c_i に属する符号語は全て値 d_i を表し，誤り訂正能力 t ビットを持ち，値 d_i を表す符号語 v_k から値 d_j を表す符号語 v_l へは書き込みビット数 S ビット以下で遷移できる．

例 1. (7, 4, 3) ハミング符号を考える．(7, 4, 3) ハミング符号は符号語 $\{v_0, \ldots, v_{15}\}$ から構成される．最小ハミング距離は 3 であるため，$t = 1$ ビットの誤り訂正能力を持つ．

定数 $S = 3$ を与え，S-バウンドグラフを生成する．生成した S-バウンドグラフを図 2(a) に示す．図 2(a) での各ノードは (7, 4, 3) ハミング符号の符号語を表す．$H(v_0, v_1) = 3 \leq 3$ であるため，ノード v_0, v_1 間にはエッジが存在する．$H(v_0, v_{15}) = 7 \geq 3$ であるため，ノード v_0, v_{15} 間にはエッジが存在しない．

S-バウンドグラフを元にクラスタグラフを生成する．生成したクラスタグラフを図 2(b) に示す．図 2(b) では 8 つのクラスタ $c_0, \ldots, c_7$ があり，各クラスタは 2 つのノードを持つ．例えば，クラスタ c_0 は $\{v_0, v_{15}\}$ を持ち，クラスタ c_1 は $\{v_1, v_{14}\}$ を持つ．S-バウンドグラフでは v_0, v_1 と v_{14}, v_{15} はそれぞれエッジで結ばれているため，S ビット反転条件を満足する．そのため，エッジ $e' = (c_0, c_1)$ を結ぶ．図 2(b) のクラスタグラフは完全グラフとなる．

次に，3 ビットの 8 つの値 $d_0, \ldots, d_7$ を考える．各値は $d_0 = 000, d_1 = 001, \ldots, d_7 = 111$ となる．値 d_i を $c_i (0 \leq i \leq 7)$ に割り当てる．例えば，クラスタ c_0 の符号語 $v_0 = 0000000, v_{15} = 1111111$ はどちらも値 $d_0 = 000$ を表す．また，クラスタ c_1 の符号語 $v_1 = 0001011, v_{14} = 1110100$ はどちらも値 $d_1 = 001$ を表す．

値 $d_0 = 000$ を表す符号語 $v_0 = 0000000$ がメモリに書き込まれているとする．値 $d_1 = 001$ をメモリに保存するとき，$v_1 = 0001011$ と $v_{14} = 1110100$ を書き込む 2 通りの書き込み方が存在する．この場合，$v_1 = 0001011$ を書き込むと値 d_0 を表す符号語 v_0 から値 d_1 を表す符号語 v_1 の遷移に 3 ビットの書き込みで済む．

以上より，生成した符号は 3 ビットの値を 7 ビットの符号語にエンコードする．この符号は 1 ビットの誤り訂正能力を持ち，全ての値の遷移は適切に符号語を選択することで 3 ビットで書き換えができる．

5 提案アルゴリズム

4 章で全てのクラスタの組み合わせが S ビット反転条件を満足することで誤り訂正能力 t を維持しつつ，書き込みビット数を S ビット以下に制約できる符号を生成できることを示した．本章では，クラスタリング条件を満足し，全てのクラスタの組み合わせが S ビット反転条件を満足する誤り訂正符号の符号語のクラスタリング手法を示す．

$(n, k_p, 2t+1)$ 線形組織誤り訂正符号の符号語をクラスタリングすることを考える．$(n, k_p, 2t+1)$ 線形組織誤り訂正符号の符号語を $\{v_0, \ldots, v_{2^{k_p}-1}\}$ とする [11]．符号語には n ビットの符号語 $11\cdots11$ が含まれていることとする．この符号では，k_p ビットのメッセージ $m_i (0 \leq i \leq 2^{k_p} - 1)$ がエンコードされて n ビットの符号語 v_i となる．エンコードとはメッセージを誤り訂正符号の生成行列を用いて誤り訂正符号の符号語に変換することである．線形符号とはその符号の 2 つの符号語を XOR して得られるビット列が符号語となる符号である．組織符号とは符号語がメッセージと冗長なビット列で構成された符号である．誤り訂正能力は t ビットである．

提案アルゴリズムでは，始点ベクトル a と情報ベクトル x を用いてクラスタリングする．a, x はどちらも $(n, k_p, 2t+1)$ 誤り訂正符号の符号語である．a を始点ベクトル，x を情報ベクトルと呼ぶ．始点ベクトルの集合を $A = \{a_0, \ldots, a_{2^r-1}\}$，情報ベクトルの集合を $X = \{x_0, \ldots, x_{2^{k_r}-1}\}$ とする．ここで，r を冗長量としたとき，$k_r = k_p - r$ とする，A，X をうまく設定することで全ての符号語は $v = a \oplus x$ として表せる．

5.1 始点ベクトルの生成法

冗長量を $r \geq 1$，情報量を $k_r = k_p - r$ と表す．r, k_r は $k_r \equiv 0 \pmod{r}$ とする．n ビットの始点ベクトルの集合を $A = \{a_0, \ldots, a_{2^r-1}\}$ と表し，k_p ビットの始点メッセージの集合を $M^a = \{m_0^a, \ldots, m_{2^r-1}^a\}$ とする．$m_i^a(j)$ を m_i^a の j 番目のビットとする．つまり，$m_i^a(0)$ は m_i^a の LSB，$m_i^a(k_p - 1)$ は m_i^a の MSB を表す．始点ベクトル a_i は始点メッセージ m_i^a をエンコードして得られる．

始点ベクトルの集合 A は以下の操作により得られる．

Step 1: $0 \leq i < 2^{r-1}$ に対して，m_i^a の上位 r ビットは i を 2 進数表記したものとする．例えば $k_p = 6, r = 3$ のとき，$m_0^a = \underline{000}$***, $m_1^a = \underline{001}$***, $m_2^a = \underline{010}$***, $m_3^a = \underline{011}$*** となる．ただし，*は未定義である．

Step 2: $0 \leq i < 2^{r-1}$ に対して，Step (2-1), Step (2-2) を実行する．

Step (2-1): $W_H(i)$ が偶然ならば，メッセージ m_i^a の下位 k_r の各 j ビット目は下式となる．

$$m_i^a(j) = m_i^a(k_r + \lfloor j \times r / k_r \rfloor). \quad (1)$$

Step (2-2): $W_H(i)$ が奇数ならば，メッセージ m_i^a の下位

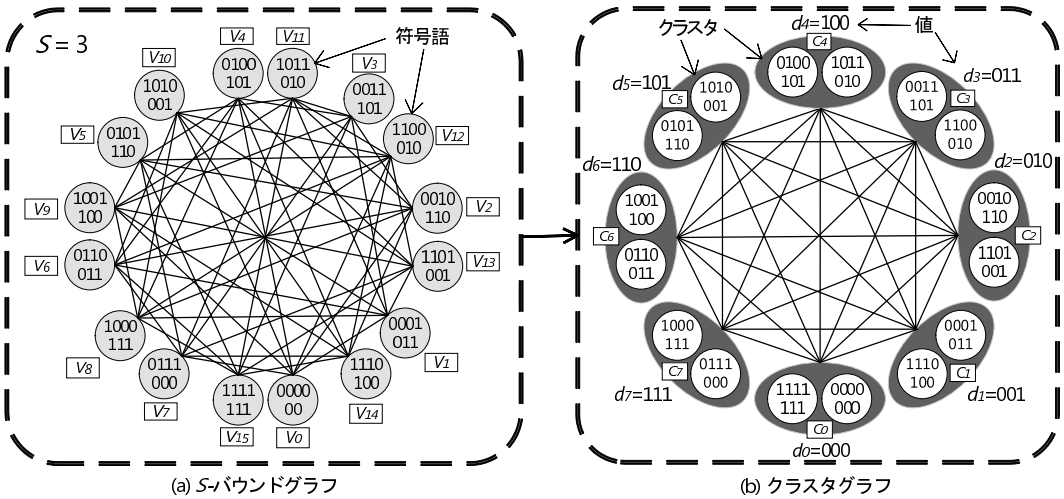

図 2　S-バウンドグラフ $G=(V,E)$(a) とクラスタグラフ $G'=(C,E')$(b).

k_r の各 j ビット目は下式となる.

$$m_i^a(j) = \overline{m_i^a(k_r + \lfloor j \times r/k_r \rfloor)}. \tag{2}$$

Step 3:　$2^{r-1} \le i \le 2^r - 1$ に対して，メッセージ m_i^a は以下のようになる.

$$m_i^a = \overline{m_{2^r-1-i}^a}. \tag{3}$$

以上の操作で得られた k_p ビットのメッセージ m_i^a をエンコードすることで n ビットの始点ベクトル a_i が得られる.

例 2. $k_p = 6$, $r = 2$ とする. $k_r = k_p - r = 4$ となる. 6 ビットのメッセージ m_i^a を上に示した手法により生成する.

Step 1 より，m_0^a, m_1^a の上位 2 ビットは以下のようになる.

$$\begin{aligned} m_0^a &= \underline{00}{*}{*}{*}{*} \\ m_1^a &= \underline{01}{*}{*}{*}{*} \end{aligned} \tag{4}$$

ただし，*は未定義である.

Step 2 より，m_0^a, m_1^a の下位 4 ビットは以下のようになる.

$$\begin{aligned} m_0^a &= 00\underline{0000} \\ m_1^a &= 01\underline{1100} \end{aligned} \tag{5}$$

Step 3 より，m_2^a, m_3^a は m_0^a, m_1^a を元に以下のようになる.

$$\begin{aligned} m_2^a &= \underline{100011} \\ m_3^a &= \underline{111111} \end{aligned} \tag{6}$$

以上の操作で得られた 6 ビットのメッセージ m_i^a をエンコードすることで n ビットの始点ベクトルが得られる.

5.2　情報ベクトルの生成法

n ビットの情報ベクトルの集合を $X = \{x_0, \ldots, x_{2^{k_r}-1}\}$ と表し，k_p ビットの情報メッセージの集合を $M^x = \{m_0^x, \ldots, m_{2^{k_r}-1}^x\}$ とする. $m_i^x(j)$ を m_i^x の j 番目のビットとする. つまり，$m_i^x(0)$ は m_i^x の LSB，$m_i^x(k_p-1)$ は m_i^x の MSB を表す. 情報ベクトル x_i は情報メッセージ m_i^x をエンコードして得られる.

情報ベクトルの集合 X は以下の操作により得られる.

Step 1:　m_i^x の上位 r ビットは 0 を 2 進数表記したものとする.

Step 2:　m_i^x の下位 k_r ビットは i を 2 進数表記したものとする.

以上の操作で得られた k_p ビットのメッセージ m_i^x をエンコードすることで n ビットの情報ベクトル x_i が得られる.

例 3. $k_p = 6$, $r = 2$ とする. $k_r = k_p - r = 4$ となる. 6 ビットのメッセージ m_i^x を上に示した手法により生成する.

Step 1 より，$m_0^x, \ldots, m_{15}^x$ の上位 2 ビットは以下のようになる.

$$\begin{aligned} m_0^x &= \underline{00}{*}{*}{*}{*} \\ m_1^x &= \underline{00}{*}{*}{*}{*} \\ m_2^x &= \underline{00}{*}{*}{*}{*} \\ &\vdots \\ m_{15}^x &= \underline{00}{*}{*}{*}{*} \end{aligned} \tag{7}$$

ただし，*は未定義である.

Step 2 より，$m_0^x, \ldots, m_{15}^x$ の下位 4 ビットは以下のようになる.

$$\begin{aligned} m_0^x &= 00\underline{0000} \\ m_1^x &= 00\underline{0001} \\ m_2^x &= 00\underline{0010} \\ &\vdots \\ m_{15}^x &= 00\underline{1111} \end{aligned} \tag{8}$$

以上の操作で得られた 6 ビットのメッセージ m_i^x をエンコードすることで n ビットの情報ベクトル x_i が得られる.

5.3　始点ベクトルと情報ベクトルの性質

本節では始点ベクトルと情報ベクトルの性質を示す.

定理 3. 前節で示した手法によって始点ベクトルの集合 A と情報ベクトルの集合 X を生成する. 全ての符号語 $v_i \in \{v_0, \ldots, v_{2^{k_p}-1}\}$ は $v_i = a \oplus x (a \in A, x \in X)$ によって一意に表せる.

証明 3. $a \oplus x$ で全ての符号語が表され，$a \oplus x$ は $2^r \times 2^{k_r} = 2^{r+k_r} = 2^{k_p}$ 個の組み合わせを持ち，誤り訂正の符号語数も 2^{k_p} 個であるため，上記定理が成立する. □

定数 $S = \lfloor n/2 \rfloor$ を与え，S-バウンドグラフ $G=(V,E)$ を生成する. r を冗長量として $k_r = k_p - r$ とする. ノードの集合 V からクラスタ c_i $(i = 0, \ldots, (2^{k_r}-1))$ を以下のように生成する.

$$c_i = \{a_0 \oplus x_i, \ldots, a_{2^r-1} \oplus x_i\} \tag{9}$$

表 1　符号の性質の比較 1 (情報量 4 の場合).

	符号長	誤り訂正能力	最大書き込みビット数	最小書き込みビット数	平均書き込みビット数
(7, 4, 3) ハミング符号	7	1	7	3	3.50 (1.00)
(9, 4, 3, 1)-REC	9	1	4	3	3.25 (0.93)
(10, 4, 3, 2)-REC	10	1	4	3	3.12 (0.89)
(12, 4, 3, 4)-REC	12	1	4	3	2.93 (0.83)

表 2　符号の性質の比較 2 (情報量 8 の場合).

	符号長	誤り訂正能力	最大書き込みビット数	最小書き込みビット数	平均書き込みビット数
(15, 7, 5)-BCH 符号 + (5, 1, 5) 反復符号	20	2	20	5	10.0 (1.00)
(23, 8, 5, 1)-REC	23	2	11	5	8.92 (0.89)
(25, 8, 5, 2)-REC	25	2	12	5	8.67 (0.86)
(29, 8, 5, 4)-REC	29	2	12	5	7.80 (0.78)
(37, 8, 5, 8)-REC	37	2	10	5	7.18 (0.71)

これらのクラスタはクラスタ条件 (C1)–(C3) を満足する．

クラスタの集合を $C = \{c_0, \ldots, c_{2^{k_r}-1}\}$ とする．$c_i, c_j (i \neq j)$ が S ビット反転条件 (F1), (F2) を満たすとき，c_i, c_j をエッジ $e' = (c_i, c_j)$ で結ぶ．このようにしてクラスタグラフ $G' = (C, E')$ を新たに生成する．

定理 4. 上述のようにして生成したクラスタグラフ $G' = (C, E')$ は完全グラフとなる．

証明 4. $0 \leq i \leq 2^{r-1}$ のとき，始点ベクトル a_i は始点ベクトル a_{2^r-1-i} を反転した符号語である．この性質を用いると全てのクラスタ間で S-ビット反転条件を満足するエッジが存在するため，上記定理が成立する． □

k_r ビットの値の集合を $D = \{d_0, \ldots, d_{2^{k_r}-1}\}$ とする．クラスタ c_i に値 d_i を割り当てる．つまり，クラスタ $c_i \in C$ に属する全ての符号語は値 $d_i \in D$ を表すことにする．

定理 1, 2, 4 より，クラスタ c_i に属する全ての符号語は誤り訂正能力 t を持ち，値 d_i を表す．値 d_i を表す符号語 v_k から値 d_j を表す符号語 v_l への遷移は $\lfloor n/2 \rfloor$ 以下の書き込みビット数で済む．新たに生成した符号を $(n, k_r, 2t+1, r)$-REC 符号と呼ぶ．n は符号長，k_r は情報量，$2t+1$ は最小ハミング距離，r は冗長量を表す．

図 3 に $(n, 2, 2t+1, 2)$-REC 符号の符号空間を示す．4 つの始点ベクトル a_0, a_1, a_2, a_3 があり，4 つの長方形の中の左下の円が対応する．4 つの情報ベクトル x_0, x_1, x_2, x_3 があり，4 つの長方形の中の 4 本の矢印が対応する．全ての符号語は $a_i \oplus x_j (i = 0, 1, 2, 3), (j = 0, 1, 2, 3)$ で表せる．クラスタ $c_j (j = 0, 1, 2, 3)$ には符号語 $c_j = \{a_0 \oplus x_j, a_1 \oplus x_j, a_2 \oplus x_j, a_3 \oplus x_j\}$ が属し，全ての c_j に属する符号語 $a_i \oplus x_j \in c_j$ は値 d_j を表す．

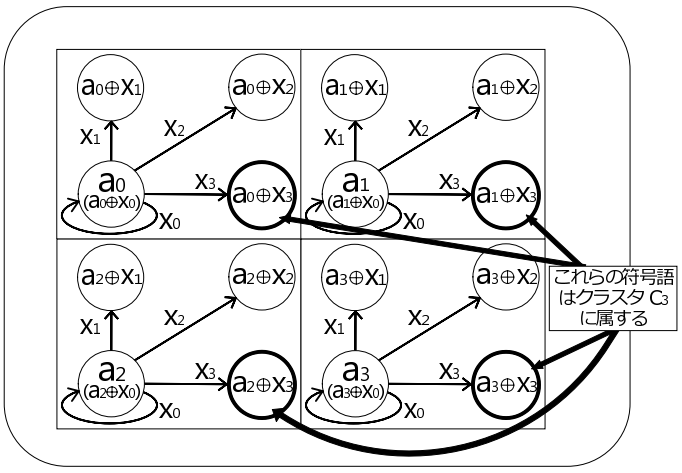

図 3　$r = 2$ のときの符号空間.

6　実験

本章では，REC 符号を用いて構成した不揮発メモリの性能を評価する．REC 符号と，一般に用いられる誤り訂正符号としてハミング符号と BCH 符号の符号性質の比較，それらの符号を用いて構成したメモリにアプリケーションを適用した結果を示す．

6.1　符号の性質

本節では REC 符号の性質を示す．表 1 に $(7, 4, 3)$ ハミング符号，$(9, 4, 3, 1)$-REC，$(10, 4, 3, 2)$-REC，$(12, 4, 3, 4)$-REC の符号の性質の比較を示す．次に，$(15, 7, 5)$-BCH 符号 [2] と (5, 1, 5) 反復符号 [5] を連結した符号を考える．この符号では 8 ビットの情報を 7 ビット +1 ビットに分けてそれぞれエンコードを行い，15 ビットと 5 ビットの符号語を連結した 20 ビットがメモリに格納されるものとする．表 2 に $(15, 7, 5)$-BCH 符号 $+(5, 1, 5)$ 反復符号，$(23, 8, 5, 1)$-REC，$(25, 8, 5, 2)$-REC，$(29, 8, 5, 4)$-REC，$(37, 8, 5, 8)$-REC の符号の性質の比較を示す．

いずれの結果も最大書き込みビット数，平均書き込みビット数ともに $\lfloor n/2 \rfloor$ 以下となっている．REC 符号において，各値はそれぞれ 2^r 個の符号語で表される．例えば，$(9, 4, 3, 1)$-REC 符号では冗長量は $r = 1$ であるため，各値はそれぞれ $2^r = 2$ 個の符号語で表される．冗長量 r を大きくするほど符号長と各値を表す符号語数は大きくなるが，最大書き込みビット数と平均書き込みビット数は小さくなる．

表 3　書き込みビット数の比較 (情報量 4 の場合).

	adpcmE	adpcmD	epicE	epicD	g721E	g721D
(7, 4, 3) ハミング符号	2,047,028 (1.00)	735,281 (1.00)	935,223 (1.00)	520,951 (1.00)	2,129,981 (1.00)	829,899 (1.00)
(9, 4, 3, 1)-REC	1,703,040 (0.83)	704,241 (0.96)	886,396 (0.95)	501,421 (0.96)	1,791,891 (0.84)	767,722 (0.93)
(10, 4, 3, 2)-REC	1,655,103 (0.81)	676,335 (0.92)	860,966 (0.92)	484,287 (0.93)	1,741,736 (0.82)	736,449 (0.89)
(12, 4, 3, 4)-REC	1,516,490 (0.74)	633,269 (0.86)	792,089 (0.85)	443,338 (0.85)	1,599,063 (0.75)	679,179 (0.82)

表 4　書き込みビット数の比較 (情報量 8 の場合).

	adpcmE	adpcmD	epicE	epicD	g721E	g721D
(15, 7, 5)-BCH 符号 + (5, 1, 5) 反復符号	2,902,865 (1.00)	1,022,642 (1.00)	1,213,777 (1.00)	670,655 (1.00)	3,030,135 (1.00)	1,140,475 (1.00)
(23, 8, 5, 1)-REC	2,492,757 (0.86)	974,215 (0.95)	1,240,812 (1.02)	695,669 (1.04)	2,625,590 (0.87)	1,073,272 (0.94)
(25, 8, 5, 2)-REC	2,303,056 (0.79)	929,465 (0.91)	1,168,265 (0.96)	649,003 (0.97)	2,426,246 (0.80)	1,012,208 (0.89)
(29, 8, 5, 4)-REC	2,146,736 (0.74)	873,537 (0.85)	1,111,116 (0.92)	629,212 (0.94)	2,264,785 (0.75)	929,645 (0.82)
(37, 8, 5, 8)-REC	2,085,224 (0.72)	811,245 (0.79)	1,038,296 (0.86)	592,397 (0.88)	2,199,998 (0.73)	888,672 (0.78)

6.2　ベンチマークアプリケーション実験

MediaBench [8] を SimpleScalar [1] を用いてトレースを取得した．対象の誤り訂正符号を用いて構成したメモリをトレースに適用して書き込みビット数を計測した．

メモリへの書き込みは同じビットは書き込まないものとする手法 [13] を想定する．表 3(a) にハミング符号を用いて構成したメモリと REC 符号を用いて構成したメモリでアプリケーションを動作させたときの結果を示す．表 4 に (15, 7, 5)-BCH 符号 +(5, 1, 5) 反復符号を用いて構成したメモリと REC 符号を用いて構成したメモリでアプリケーションを動作させたときの結果を示す．表 3 では，(12, 4, 3, 4)-REC を用いて構成したメモリがハミング符号を用いて構成したメモリと比較して最大 25.9% の書き込みビット数を削減した．表 4 では，(37, 8, 5, 8)-REC を用いて構成したメモリが (15, 7, 5)-BCH 符号 +(5, 1, 5) 反復符号を用いて構成したメモリと比較して最大 28.2% の書き込みビット数を削減した．

7　おわりに

本稿では，書き込みビット数削減と誤り訂正を同時に実現する手法として，誤り訂正符号の符号語をクラスタリングし，各クラスタに値を割り当てることで新たな符号を生成するアルゴリズムを提案した．提案アルゴリズムにより生成した REC 符号を用いて構成したメモリをアプリケーションに適用した結果，ハミング符号を用いて構成したメモリに比べ最大で 25.9%，BCH 符号 + 反復符号を用いて構成したメモリに比べ最大で 28.2% の書き込みビット数を削減した．

今後はより効率よく書き込みビット数を削減できる手法を研究する．

謝辞

本研究の一部は，総務省 SCOPE の支援による．

参考文献

[1] T. Austin, E. Larson, and D. Ernst, “SimpleScalar: an infrastructure for computer system modeling,” *IEEE Trans. Computers*, vol. 35, issue 2, pp. 59–67, 2002.

[2] R. C. Bose and D. K. Ray-Chaudhuri, “On a class of error correcting binary group codes,” *Information and Control*, vol. 3, pp. 68–79, 1960.

[3] J. Chen, R. C. Chiang, H. H. Huang, and G. Venkataramani, “Energy-aware writes to non-volatile main memory,” *ACM SIGOPS Operating Systems Review*, vol. 45, no. 119, pp. 48–52, 2011.

[4] S. Cho and H. Lee, “Flip-N-Write: a simple deterministic technique to improve PRAM write performance, energy and endurance,” in *Proc. MICRO 2009*, pp. 347–357, 2009.

[5] S. Gravano, M. C. Doggett, and P. J. Mcdougall, “Comparison of a cyclic code and a repetition code with the same code rate in the presence of single-bit errors,” *Int. Journal of Electronics*, vol. 67, no. 4, pp. 495–502, 1989.

[6] R. W. Hamming, “Error detecting and error correcting codes,” *Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, 1950.

[7] T. Kojo, M. Tawada, M. Yanagisawa, and N. Togawa, “A write-reducing and error-correcting code generation method for non-volatile memories,” in *Proc. APCCAS 2014*, pp. 304–307, 2014.

[8] C. Lee, M. Potkonjak, and W. H. Mangione-Smith, “MediaBench: a tool for evaluating and synthesizing multimedia and communications systems,” in *Proc. MICRO 1997*, pp. 330–335, 1997.

[9] K. Mohanram and S. Rixner, “Context-independent codes for off-chip interconnects,” in *Proc. Workshop on Power-Aware Computer Systems*, pp. 107–119, 2004.

[10] H. Noguchi, K. Ikegami, N. Shimomura, T. Tetsufumi, J. Ito, and S. Fujita, “Highly reliable and low-power nonvolatile cache memory with advanced perpendicular STT-MRAM for high-performance CPU,” in *Proc. VLSI Technology 2014*, pp. 1–2, 2014.

[11] H. Ohnsorge, “Linear systematic code encoding and detecting devices,” U.S. Patent 3 512 150, May 12, 1965.

[12] M. Tawada, S. Kimura, M. Yanagisawa, and N. Togawa, “A bit-write reduction method based on error-correcting codes for non-volatile memories,” in *Proc. ASP-DAC 2015*, pp. 496–501, 2015.

[13] P. Zhou, B. Zhao, J. Yang, and Y. Zhang, “Energy reduction for STT-RAM using write termination,” in *Proc. ICCAD 2009*, pp. 264–268, 2009.