リードソロモン符号に基づいた マルチレベルセル不揮発性メモリ書き込み削減

多和田　雅師[†1,a)]　柳澤　政生[†1]　戸川　望[†1]

概要：近年 IoT デバイスの普及に伴いノーマリーオフ可能な不揮発性メモリの需要が高まっている．特に 1 個のメモリセルに複数の状態を割り当てるマルチレベルセル不揮発性メモリが主流である．マルチレベルセル不揮発性メモリは書き込み耐久性が低いという問題がある．この問題を解決するため誤り訂正符号に基づくメモリ冗長化を考える．情報をメモリセルをに書き込むとき，冗長化されているため書き込み方のパターンが多くなる．書き込み内容に加えて書き込み位置を情報とみなせるため，少ない書き換えメモリセルでも多くの情報を保存できる．リードソロモン符号に基づきメモリセルの状態を書き換え，書き換えるメモリセルの個数を削減し書き込み耐久性を向上させる．

キーワード：セルレベル書き込み削減, 不揮発性メモリ, 書き込み削減符号, リードソロモン符号

1. はじめに

LSI 製造技術の向上によりチップ上に微細な論理回路の搭載が可能となった．微細な論理回路により演算性能が向上するほど消費電力も増加するため，ノーマリオフコンピューティングやパワーゲーティングといった技術により消費電力を削減する必要がある．ノーマリオフコンピューティングやパワーゲーティングをシステムに実装するために不揮発性メモリを使用することが考えられる．不揮発性メモリは電力の供給がなくても情報を保持できるためノーマリオフコンピューティングやパワーゲーティングに有効である．不揮発メモリには PCM (Phase Change Memory) や MRAM (Magnetic Random Access Memory) などの種類がある [1]．一方で，不揮発性メモリは書き込みエネルギーが大きい，書き換え回数の上限が低いといった欠点がある．この欠点を補完するためメモリに書き込むビットパターンを符号化し，書き込み量を削減する手法が研究されている [2], [3], [4], [5], [6]．書き込み前のメモリの状態と書き込む値に対応するメモリの状態を比較し，必要最小限の書き換えを行うことで書き込み量を削減することができる [7].

冗長符号化によりメモリを冗長化すると書き込みの位置も情報となり，同じ書き込み量でも与える情報量が増加する．図 1 に示すように 31 ビットのメモリに 1 ビット以下の書き込みを行う場合について考える．書き込み方は 1 ビットも書き込まない場合 1 通りと 1 ビット書き込む場合 31 通り合わせて 32 通りの書き込み方がある．1 ビット以下の書き込みにより 32 通りの書き込み方，つまり 5 ビットの情報量が保存できる．このようなビットを書き込む位置を情報とすることで本質的に書き込み量を削減する．メモリ内の値を符号語とみなすことで書き込むビット数を削減できる [2], [3].

これらの冗長符号化手法はシングルレベルセル不揮発性メモリを対象としている．シングルレベルセルはセルと呼ばれる記憶素子の単位に対し 1 ビットの状態割り当てを行うため，ビットの反転がそのまま書き込み量と考えることができた．しかし，シングルレベルセルでは不揮発性メモリはセルの集積度が低くなるためマルチレベルセルと呼ばれるセル集積度が高い方式が使用される．マルチレベルセルはセルひとつに複数の状態を割り当てる方式である．マルチレベルセルではビットと物理的な状態の遷移が対応していないため，既存のシングルレベルセル不揮発性メモリを対象とした書き込み量削減手法では高い効果を得ることができない．セルレベルでの書き込み量削減を考える必要がある．セルレベル書き込み削減手法としてビットレベル書き込み削減手法 FNW (Flip-N-Write)[2] をマルチレベルセルに適用させた MFNW (MLC FNW) が存在する [8].

マルチレベルセルのうち特に 1 セルで 8 値の情報を格納する方式をトリプルレベルセル (TLC) と呼ぶ．マルチレ

†1 現在，早稲田大学
Presently with Waseda University
a) tawada@togawa.cs.waseda.ac.jp

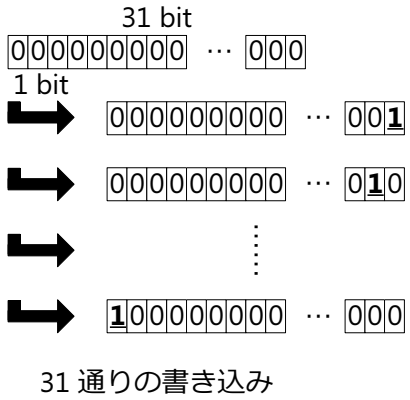

図 1 1 ビット書き込みによる 5 ビットの情報量.

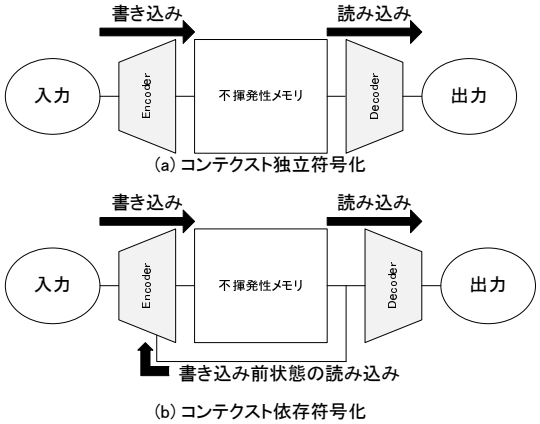

図 2 コンテクスト独立符号化とコンテクスト依存符号化.

ベルセルやトリプルレベルセルのようにセルあたりの状態数が増加すると MFNW では書き込み削減効率が下がる問題がある．この問題に対しセルレベルで書き込み削減を考える手法を研究する．本稿ではリードソロモン誤り訂正符号に基づいて情報の分散化を行いセルレベルで書き込み量を削減する手法を提案する．提案手法についてセルの状態遷移量の観点で書き込み量を評価する．

2. 不揮発性メモリの書き込み削減

冗長符号を用いた書き込み削減手法では，保存したい値を符号語に変換して不揮発性メモリに書き込む．不揮発性メモリに保存された符号語は値に変換して読みだされる．この変換を実現するエンコーダ，デコーダが必要となる．エンコーダとデコーダの構成は 2 通りの実現方法がある [9]．1 つはコンテクスト独立符号化方式であり，メモリの現状態に関わらず値に対してエンコーダの変換をする方式である．これはエンコード回路が組合せ回路で構成でき，値と符号語が 1 対 1 に対応する．メモリの内容からのフィードバックがないためレイテンシは小さくなるが，使用されないメモリ状態が存在するため情報量に無駄が発生し書き込み削減量の効率は低い．もう 1 つはコンテクスト依存符号化方式であり，メモリにすでに書き込まれている符号語に依存して値をエンコードする方式である．メモリ

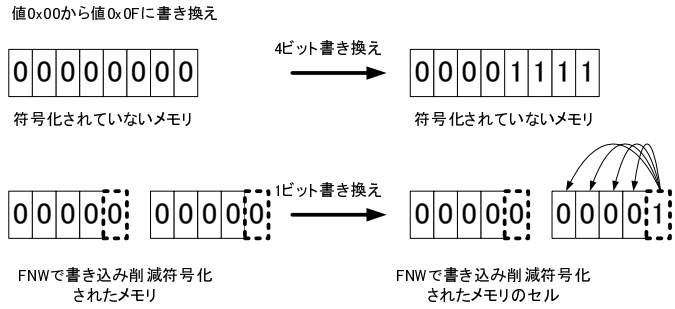

図 3 FNW[2] のメモリの動作.

の内容からフィードバックを受けるためレイテンシは大きくなるが，メモリの前状態から書き込み量の小さい次状態を選択できるため効率的に書き込み削減できる．図 2 にコンテクスト独立符号化とコンテクスト依存符号化のエンコーダとデコーダのアーキテクチャを示す．本稿では対象とするアーキテクチャは図 2 のコンテクスト依存符号化方式を対象とする．

2.1 FNW[2]

FNW はシングルレベルセル不揮発性メモリを対象とする書き込み削減手法である．値を 1 ビット冗長化して符号語とする．この 1 ビットはフリップビットと呼ばれ，他のビットに XOR することで復号できる．フリップビットが 0 のとき，デコーダにより他のビットは反転なく出力される．フリップビットが 1 のとき，デコーダにより他のビットは反転されて出力される．値 0x00 が保存されている 8 ビット長のメモリに対して値 0x0F が書き込まれることを考える．図 3 に FNW のメモリの動作を示す．冗長符号化されていないメモリでは 8 ビットの値を 8 ビットそのままで保持し，書き込みビット数は 4 ビットとなる．FNW を適用させたメモリでは，8 ビットの値を 4 ビットごとに管理し 10 ビットで保持し，下位 4 ビットすべてが反転するような書き込みに対してはフリップビット 1 ビットのみを反転させる．書き込みビット数が 4 ビットから 1 ビットに削減できたことになる．

2.2 MFNW[8]

MFNW はビットレベル書き込み削減手法 FNW をマルチレベルセル不揮発性メモリの書き込み削減に適用させた手法である．セルを 1 個冗長化してメモリに実装する．このセルはタグセルと呼ばれ，他のセルに対しビット表現で XOR することで復号できる．図 4 にセルの状態とビット表現の対応を示す．MFNW はセルが表すビット表現について，マルチレベルセル 1 個が表すビット表現を各ビットそれぞれについて FNW の符号語の一部と考える．例えば，図 4 のセル状態 0 に注目するとき，冗長化されていないメモリではビット表現で $W_0 = 0011011$ であるがセル状態 0 では $W_1 = 001101100$ と表される．これは各セルの

ビット表現	00	01	10	11	
セルの状態0	00	01	10	11	00
セルの状態1	01	00	11	10	01
セルの状態2	10	11	00	01	10
セルの状態3	11	10	01	00	11

図 4 MFNW[8] のメモリの表現.

各ビットをつなげて $W_L = 0011\underline{0}$, $W_R = 0101\underline{0}$ とみなしそれぞれを FNW の符号語と考える．同様にセル状態 1 は $W_L = 0011\underline{0}$, $W_R = 1010\underline{1}$, セル状態 2 は $W_L = 1100\underline{1}$, $W_R = 0101\underline{0}$, セル状態 3 は $W_L = 1100\underline{1}$, $W_R = 1010\underline{1}$ という FNW の符号語と考える．このようにセルの状態を割り当てることでマルチレベルセル不揮発性メモリを符号化し，最も書き込み量の小さいセル状態になるようにエンコードすることで書き込み削減する．

3. リードソロモン符号

不揮発性メモリの情報書き込みを冗長符号に変換して書き込むことで，ビットレベルで考えたとき書き込み削減が可能となることが分かった．しかし，不揮発性メモリのセルに対して複数のビットを割り当てるマルチレベルセル不揮発性メモリを考えると，ビット単位で書き込み削減をすることは必ずしもセル単位で書き込み削減することにつながらない．ビット単位で書き込み削減をするのではなくセル単位で書き込み削減するためには，誤り訂正符号もビット単位で訂正するものではなくセルに相当する単位で訂正する符号を使う必要がある．ビット単位ではなく複数のビットを含むシンボル単位で誤り訂正する符号としてリードソロモン符号がある．リードソロモン符号に注目し書き込み削減を考える．

3.1 ビットレベル書き込み削減手法の問題点

マルチレベルセル不揮発性メモリでは書き込み量を CHD (Cell Hamming Distance) で評価する [8]．CHD はビットではなくセルの変化量によって書き込み量を規定するため，1 ビット変化の 1 セル書き込みも 2 ビット変化の 1 セル書き込みと同等に評価される．ビットレベル書き込み削減手法の問題点として変更されるビット数が減少しても変更されるセル数が増加する可能性がある．図 5 にビットレベル書き込み削減手法しても書き込みセルが増加する例を示す．いま符号化されていないメモリに値 0x00 が入っていたとする．このメモリの値を 0x1F に書き換えるとき，5 ビットの書き換えが発生する．2 ビットごとに 1 セルとしてメモリセルに格納されているならばセル単位では 3 セルの書き換えとなる．同様にビットレベル書き込み削減符号

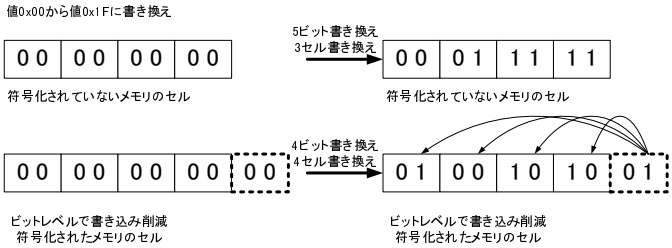

図 5 ビットレベル書き込み削減手法しても書き込みセルが増加する例.

表 1 ガロア体の各要素の表現.

べき表現	多項式表現	ビット表現
0	0	000
1	1	001
α	α	010
α^2	α^2	100
α^3	$\alpha+1$	011
α^4	$\alpha^2+\alpha$	110
α^5	$\alpha^2+\alpha+1$	111
α^6	α^2+1	101

化されたメモリ，ここでは MFNW[8] に値 0x00 が入っていたとする．このメモリの値を 0x1F に書き換えるとき，4 ビットの書き換えと 4 セルの書き換えが発生する．ビット単位では書き込み削減の効果は出ているがセル単位では書き込み量が増加している．

ビット単位で書き込み削減する符号はセル単位の書き込み量を考慮しないため，書き込みビットが食い違い，書き込みセルを増加させる．トリプルレベルセルのようにセルが表現するビットが大きくなるほどこの問題は深刻になる．

3.2 リードソロモン符号の構成

リードソロモン符号はシンボル単位で誤り訂正する符号である．リードソロモン符号の符号語はシンボルを係数とする多項式で表される．本稿におけるリードソロモン符号はシンボルの状態数 8，シンボル長 7 シンボル，情報量 3 シンボル，誤り訂正能力 2 シンボルの (7,3)RS 符号とする．シンボルを 3 ビットとし，要素数 8 のガロア体をシンボルの各状態とする．

$$\alpha^3 + \alpha + 1 = 0 \tag{1}$$

本稿では式 1 を用いてガロア体を構成する．このとき，各要素は表 1 のように表現できる．本稿では記号 “+” は 2 を法とする加算である．α^3 は式 1 より $\alpha^3 = -\alpha - 1$ であり，2 を法とするため，$\alpha^3 = \alpha + 1$ である．同様に $\alpha^4, \alpha^5, \alpha^6$ も α の 2 次多項式で表現できる．また，α^2,α,1 の係数を取り出すことでビット表現にできる．(7,3)RS 符号の符号語の生成には生成多項式 $G(x)$ を用いる．(7,3)RS 符号の生成多項式を式 2 とする．

$$G(x) = x^4 + \alpha^2 x^3 + \alpha^5 x^2 + \alpha^5 x + \alpha^6 \tag{2}$$

(7,3)RS 符号の符号語は生成多項式 $G(x)$ による剰余が 0 となる．例えば 6 次多項式 $x^6+\alpha^2x^5+\alpha^5x^4+\alpha^5x^3+\alpha^6x^2$ は (7,3)RS 符号の符号語であり，生成多項式 $G(x)$ による剰余が 0 となる．この符号語の係数のシンボルを列挙すると，$1,\alpha^2,\alpha^5,\alpha^5,\alpha^6,0,0$ となり，それぞれのシンボルをビット表現に変換すると，001100111111101000000 となる．この 21 ビットが (7,3)RS 符号の符号語の一例である．同様に生成多項式 $G(x)$ による剰余が 0 となる 21 ビットのビットパターンを作り出すことで (7,3)RS 符号の符号語を生成することができる．得られた符号語は誤りが混入しても 2 シンボルまでならば誤り訂正できる．例えば語 001100111111101000001 に対し，符号語なのか誤り語なのかを判定するとき，語を 6 次多項式とみなすと $x^6+\alpha^2x^5+\alpha^5x^4+\alpha^5x^3+\alpha^6x^2+1$ となる．これは生成多項式 $G(x)$ による剰余が 0 とならないため誤り語であることが確認できる．誤り値多項式と誤り位置多項式を解くことで符号語 001100111111101000000 に誤りベクトル 000000000000000000001 が混入したものであることがわかる．(7,3)RS 符号を組織符号とみなすと情報量 3 シンボルに相当する上位 9 ビットが符号化前の値を意味する．例えば符号語 001100111111101000000 は値 001100111 を符号化したものであるとみなせる．

4. 提案手法

(7,3)RS 符号の符号語を A_i とする．A_i は上位 9 ビットに対して一意であり，512 個ある．さらに (7,3)RS 符号によって訂正可能な誤りベクトルを X_j とする．(7,3)RS 符号は 2 シンボル誤り訂正可能なので X_j は高々 2 シンボルが 0 でない 21 ビットのベクトルである．そのような X_j の数は，誤りシンボル数が 0 シンボルのものが ${}_7C_0$ 個，誤りシンボル数が 1 シンボルのものが ${}_7C_1\times7$ 個，誤りシンボル数が 2 シンボルのものが ${}_7C_2\times7^2$ 個であり，計 1079 個ある．A_i+X_j は重複なく 512×1079 通りの組合せがあるが，誤り訂正によって一意に A_i と X_j に分離可能である．リードソロモン符号では A_i+X_j の示す情報は A_i と同値であるとみなす．ここで，A_i+X_j の示す情報は X_j と同値であるとみなすような新たな符号を考える．そのような符号は 1079 通りの情報を持ち，1 つの情報に対し 512 通りの冗長性がある．図 6 に誤りベクトルを情報とみなす符号の構造を示す．1 シンボルが異なることをハミング距離 1 としてグラフを構成すると，誤り訂正符号語を中心とする誤り訂正可能な領域を考えることができる．誤り訂正可能な領域は中心とする符号語によらず一定であるため，同じ誤りベクトルを持つ誤り語が存在する．どの符号語を中心としてもある特定の情報を持つ誤りベクトルが存在することになる．

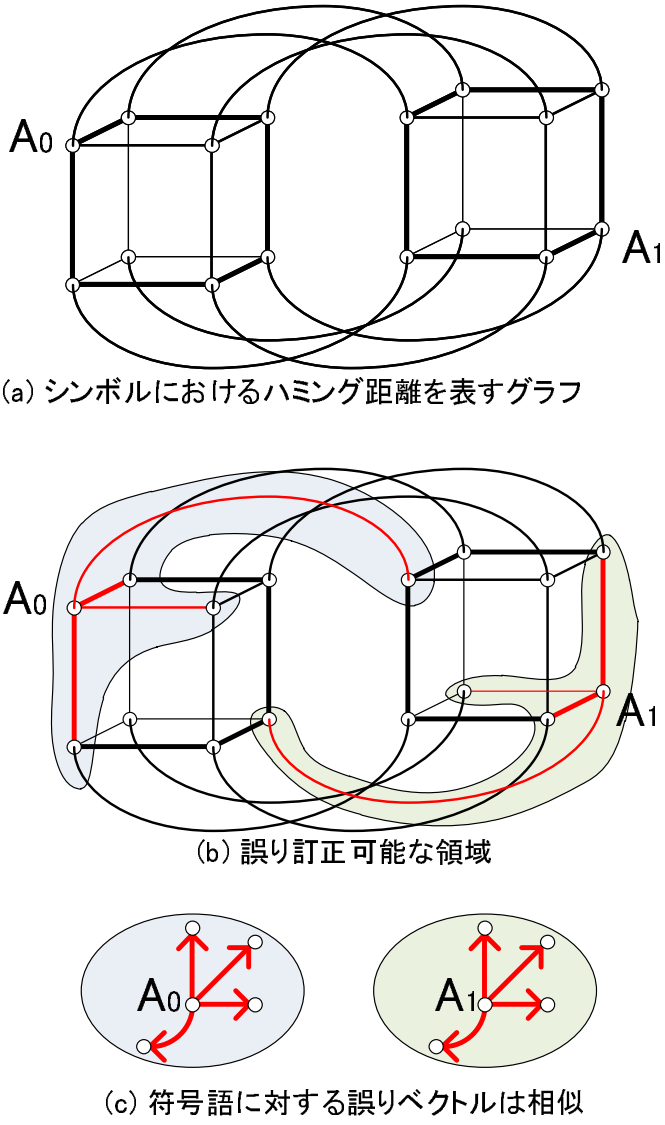

図 6 誤り訂正符号の誤りベクトルを情報と見なす新たな符号の生成.

4.1 符号の構成

リードソロモン符号はパラメータ変数 m を用いて，シンボルの状態数 $S=2^m$，シンボル長 $n_r=2^m-1$ シンボル，情報量 k_r シンボル，誤り訂正能力 $t=\frac{n_r-k_r}{2}$ シンボルと表すことができる．このとき誤り訂正可能な誤りベクトル数は $\sum_{i=0}^{t}{}_nC_i\times S^i$ 通りである．符号語を A_i，誤り訂正可能な誤りベクトルを X_j とする．提案符号のシンボル長 $n_p=n_r$ シンボル，情報量 k_p ビットとする．$2^{k_p}<\sum_{i=0}^{t}{}_nC_i\times S^i$ をみたす．提案手法による符号の構成は値と誤りベクトルの対応と誤りベクトルと符号語の対応の 2 段階で構成される．

値と誤りベクトルの対応 提案符号の値 0 から 2^{k_p} をリードソロモン符号の訂正可能な誤りベクトル X_j と対応させる．誤り訂正可能な誤りベクトル数は $\sum_{i=0}^{t}{}_nC_i\times S^i$ 通りであり，$2^{k_p}<\sum_{i=0}^{t}{}_nC_i\times S^i$ なので値に対応づけされない誤りベクトルが存在する．値に対応づけされない誤りベクトルは今後考慮し

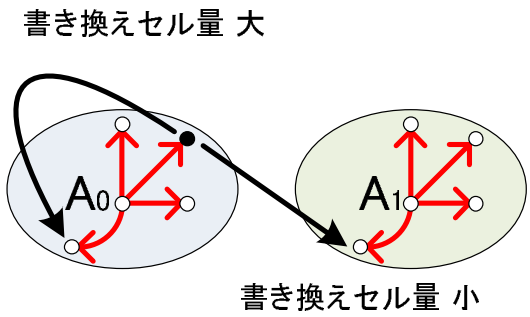

図 7 符号化による書き込み量の削減.

表 2 値と誤りベクトルの対応.

値	誤りベクトル (多項式表現)
0	0
1〜8	$\alpha^i (i = 0, 1, \ldots, 6)$
9〜16	$\alpha^i x (i = 0, 1, \ldots, 6)$
...	...
49〜56	$\alpha^i x^6 (i = 0, 1, \ldots, 6)$
57〜105	$\alpha^i x^6 + \alpha^j (i, j = 0, 1, \ldots, 6)$
106〜154	$\alpha^i x^6 + \alpha^j x (i, j = 0, 1, \ldots, 6)$
...	...
197〜245	$\alpha^i x^6 + \alpha^j x^3 (i, j = 0, 1, \ldots, 6)$
246〜255	$\alpha^i x^6 + \alpha^j x^4 (i, j = 0, 1, \ldots, 6)$

ない.

誤りベクトルと符号語の対応 誤りベクトル X_j に対して任意の誤り訂正符号語 A_i を用いて $A_i + X_j$ を対応付ける. X_j と $A_i + X_j$ は 1 対多に対応づけされる. $A_i + X_j$ は提案手法による符号の符号語である.

2 段階の対応づけを経て値と符号語が 1 対多に対応づけされた.

4.2 符号の性質

不揮発性メモリに値が符号化されて書き込まれるときに提案手法による符号では書き込み量が削減されることが期待される. 図 7 に符号化による書き込み量の削減の概念を示す. 書き込みたい値をそのまま用いて書き込みすると, 値と値の CHD がそのまま書き込み量となる. 一方, 書き込みたい値を冗長符号化すると対応する誤りベクトルを持つ符号語が複数あるため, 書き込み量が少ない符号語を選択することで書き込み量を削減できる. いま不揮発性メモリに符号語 $A_{i_0} + X_{j_0}$ が書き込まれているとして, 新たに値 D_{j_1} を書き込むことを考える. 値 D_{j_1} に対応する誤りベクトルを X_{j_1} とすると, 符号語 $A_i + X_{j_1}$ を書き込めばよい. 符号語 $A_i + X_{j_1}$ には i に自由度がある. i=i_0 とすると $A_{i_0} + X_{j_0}$ に対し $A_{i_0} + X_{j_1}$ を書き込むことになり書き込みベクトルは $X_{j_0} + X_{j_1}$ である. i=i_1 とすると $A_{i_0} + X_{j_0}$ に対し $A_{i_1} + X_{j_1}$ を書き込むことになり書き込みベクトルは $A_{i_0} + A_{i_1} + X_{j_0} + X_{j_1}$ である. ここで i をうまく選ぶことで書き込みベクトルによるセル書き込み量を削減することができる. もし $A_{i_0} + X_{j_0} + X_{j_1}$ が誤り訂正可能ならば $A_{i_0} + X_{j_0} + X_{j_1} = A_i + X_w$ となる A_i,X_w を分離することができる. このとき $(A_{i_0} + X_{j_0}) + X_w = A_i + X_{j_1}$ なので X_w は書き込みベクトル, $A_i + X_{j_1}$ は書き込む符号語となる. X_w は誤り訂正可能な誤りベクトルなので書き込み量が t シンボル以下であることが保証される. もし $A_{i_0} + X_{j_0} + X_{j_1}$ が誤り訂正不可能ならば書き込みベクトル W に用いて $(A_{i_0} + X_{j_0}) + W = A_i + X_{j_1}$ となる. このとき t シンボルより多くのセル書き込みが必要となる.

5. 評価実験

リードソロモン誤り訂正符号に基づいて情報の分散化を行いセルレベルで書き込み量を削減する手法の有効性を検証する. 対象アーキテクチャは 8 ビットの値を保存するトリプルレベルセル不揮発性メモリとする. 比較対象は符号化されていないメモリ, MFNW をトリプルレベルセルに適用させた手法, 提案手法である. 任意の 8 ビットの値から任意の 8 ビットの値に保存する値を書き換えたときの CHD を計測する.

(7,3)RS 符号をもとに提案手法を用いて新たな符号を作成する. (7,3)RS 符号では誤り訂正可能な誤りベクトルが 1079 通りある. よって新たな符号の情報量 k_p は最大 10 ビットとなる. 不揮発性メモリは 8 ビットの値を保持できればよいので最小限になるように値と誤りベクトルを対応させる. 表 2 に値と誤りベクトルの対応を示す. (7,3)RS 符号の符号語は生成多項式 $G(x)$ によって生成される. 例えば符号語 $x^6 + \alpha^2 x^5 + \alpha^5 x^4 + \alpha^5 x^3 + \alpha^6 x^2$ は値 0 を表す.

メモリ符号化手法と書き込みセル数 (CHD) の関係を表 3 に示す. 提案手法は符号化されていないメモリに比べ 3.5%, MFNW をトリプルレベルセルに適用した手法に比べ 1.5%の書き込みセル数削減を実現した. (7,3)RS 符号の誤り訂正可能な語の割合は 26%である. 74%の語は誤り訂正できず, セル書き込み量削減に寄与していない. 表 2 に示した値と誤りベクトルの対応を変化させることでよりセル書き込み量を削減できると考えられるため今後の課題とする.

6. おわりに

本稿ではリードソロモン誤り訂正符号に基づいて情報の分散化を行いセルレベルで書き込み量を削減する手法を提案した. 提案手法についてセルの状態遷移量の観点で書き込み量を評価した. 提案手法は符号化されていないメモリに比べ 3.5%, MFNW をトリプルレベルセルに適用した手法に比べ 1.5%の書き込みセル数削減を実現した. 今後はエンコーダやデコーダの面積・消費電力を評価する.

表 3 メモリ符号化手法と書き込みセル数.

	シンボル長	書き込みシンボル数	発生回数	総書き込みセル数	平均書き込みセル数
符号化されていないメモリ	3	0	256	163840	2.5
		1	4352		
		2	46592		
		3	112896		
MFNW (TLC)	4	0	256	160512	2.449
		1	6144		
		2	27136		
		3	27904		
		4	4096		
提案手法	7	0	256	158029	2.411
		1	3818		
		2	30175		
		3	31287		

謝辞

本研究の一部は，科研費 (課題番号 16K16028)，早稲田大学理工総研若手研究者支援事業「アーリーバードプログラム」の研究費による．

参考文献

[1] 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.3, pp.48–52, 2012.

[2] S. Cho and H. Lee, “Flip-N-Write: a simple deterministic technique to improve PRAM write performance, energy and endurance,” *Proc. MICRO-42*, pp.347–357, 2009.

[3] M. Tawada, S. Kimura, M. Yanagisawa, and N. Togawa, “ECC-based bit-write reduction code generation for non-volatile memory,” IEICE transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol.98, no.12, pp.2494–2504, 2015.

[4] A. Alsuwaiyan and K. Mohanram, “An offline frequent value encoding for energy-efficient MLC/TLC non-volatile memories,” *Proc. GLVLSI 2016*, pp.403–408, 2016.

[5] P.M. Palangappa and K. Mohanram, “CompEx: Compression-expansion coding for energy, latency, and lifetime improvements in MLC/TLC NVM,” *Proc. HPCA 2016*, pp.90–101, 2016.

[6] S. Swami and K. Mohanram, “E3R: energy efficient error recovery for multi/triple-level cell non-volatile memories,” *Proc. VLSID 2016*, pp.373–378, 2016.

[7] P. Zhou, B. Zhao, J. Yang, and Y. Zhang, “Energy reduction for stt-ram using early write termination,” *Proc. ICCAD 2009*, pp.264–268, 2009.

[8] A. Alsuwaiyan and K. Mohanram, “MFNW: A Flip-N-Write architecture for multi-level cell non-volatile memories,” *Proc. NANOARCH 2015*, pp.13–18, 2015.

[9] K. Mohanram and S. Rixner, “Context-independent codes for off-chip interconnects,” Power-Aware Computer Systems, pp.107–119, Springer, 2005.