TLC NAND 型フラッシュメモリの高信頼化/データ圧縮向け多元ハフマン符号の提案

出口 慶明1　渡邉 光1　竹内 健1

Triple Level Cell (TLC) NAND 型フラッシュメモリの信頼性を向上するために、多元ハフマン符号を用いた高信頼化/データ圧縮手法を提案する。TLC NAND 型フラッシュメモリではしきい値電圧を 8 状態に分けてデータを保存するが、この 8 状態に合わせた 8 元ハフマン符号を用いることで高信頼なしきい値電圧分布を生成することを提案する。さらに、7 元又は 6 元ハフマン符号を用いることで、最大 90 倍のデータ保持時間を許容できることを実測した。

1. はじめに

Triple Level Cell (TLC) NAND 型フラッシュメモリは、大容量、低コストを実現するため 1 つのメモリセルにつき 3bit を保存する。データはしきい電圧を 8 つの状態に分けることで保存され、ここでは各しきい値電圧状態をそれぞれ、Erase ステートから G ステートと呼ぶ（図 1）。

TLC NAND 型フラッシュメモリでは、データ保持時間の増加とともにエラーが発生するため信頼性に問題がある。特に、メモリセル内の電界が異なるためステート毎の信頼性が異なり、高いステートほど信頼性が低い（図 2）。本研究では、データ圧縮手法として広く使われているハフマン符号を応用し、データ圧縮と同時に信頼性の高いセルを増やす多元ハフマン符号を提案する。

2. 8 元ハフマン符号を用いた高信頼化手法

ハフマン符号では、データの発生頻度に応じてハフマン木を生成しデータを圧縮する。従来の 2 元ハフマン符号では、データを 2 つずつ繋げてハフマン木を生成する（図 3）。発生頻度が最も低い 2 つのデータをノード（四角）として接続し、新たにノード（丸）を生成する。その後、生成されたノードも含め、最も発生頻度の低いノード同士を接続してハフマン木を生成していく。全てのノードを繋げてハフマン木が完成した後、各枝に対して 2 元のデータ（"0" 又は "1"）を割り当てる。データ変調の際は、各枝に割り当てられた各ビットを辿ることでデータを変調し圧縮することができる。図 3 の例では、割り当てられた A~E のデータのうち、発生頻度が最も低い A は"111"の 3bit として保存され、発生頻度が最も高い E は"00"の 2bit として保存される。このように、発生頻度の高いデータのビット長を短くすることによってデータ圧縮を実現する。

図 4 に提案の 8 元ハフマン符号を示す。この手法では、各枝に直接 TLC NAND 型フラッシュメモリの各ステートを割り当てる。さらに、割り当てる際

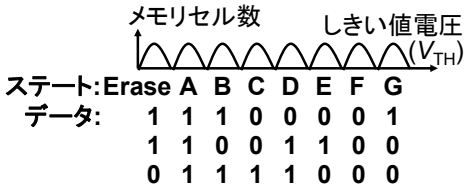


図 1 TLC NAND 型フラッシュメモリのしきい値電圧分布と各データの割り当て

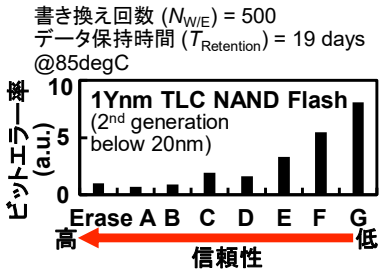


図 2 各ステートのビットエラー率実測結果[1]

に各ノードの発生頻度と各ステートの信頼性を考慮する。具体的には、同時に接続する 8 つのデータを発生頻度順に並び替え、発生頻度の高い枝に対して高信頼性なステートを割り当てる。このプロセスを繰り返してハフマン木を生成することで、データ圧縮と高信頼化が同時に実現可能となる。図 5 に、テキストファイルを提案手法で変調し実測した結果を示す。この結果では、Erase ステートのメモリセル数は全体の 27.9%、G ステートでは 6.8%となった。一般的に用いられているランダムデータを保存する場合、各ステートのセル数は 12.5%であるため、提案手法により高信頼なセルを増やし、低信頼なセルを減らすことができたことがわかる。また、圧縮率は従来の 2 元ハフマン符号で 0.572、提案手法で 0.605 であり、従来手法と同様に圧縮できてい

1 中央大学理工学研究科

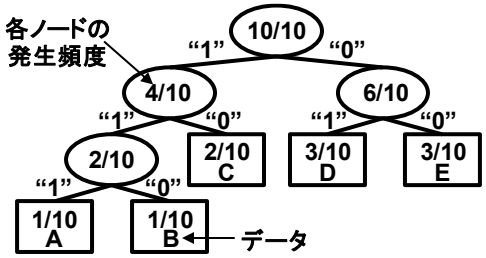


図 3 従来のハフマン符号[2]

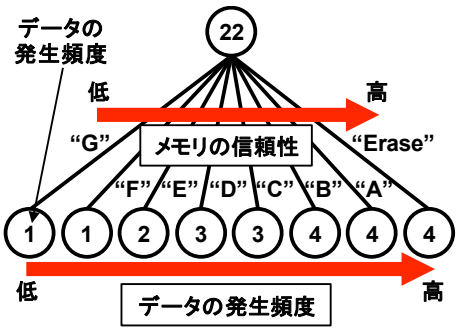
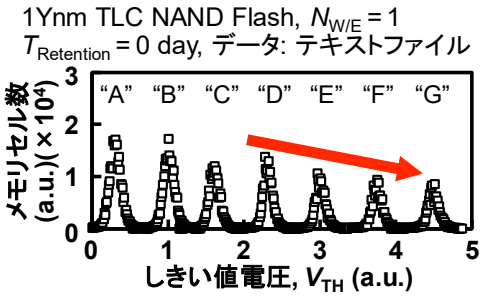


図 4 提案の 8 元ハフマン符号[1]



た。圧縮率は僅かに悪化しているが、これは枝の増加により圧縮に対して最適でなくなったことが原因である。

3. 7, 6 元ハフマン符号を用いた高信頼化手法

更なる高信頼化のため、本研究では 7、6 元ハフマン符号を用いた提案も行う。従来の高信頼化手法として *n*-out-of-8 level cell (*n*LC)と呼ばれる手法がある[3]。この手法は、TLC NAND 型フラッシュメモリの 8 つのステートのうち、1 つ又は 2 つのステートを取り除くことで信頼性を向上する手法である。本研究ではこれをハフマン符号によって実現する。7 元ハフマン符号では F ステート、6 元ハフマン符号では D と F ステートを枝に割り当てずハフマン木を生成する。また、7 元ハフマン符号で F ステートを取り除くことで G ステートの信頼性が向上することを考慮し、図 6(a)のようにステートを割り当てる。同様に、6 元ハフマン符号では E と G の信頼性向上を考慮し図 6(b)のように割り当てる。

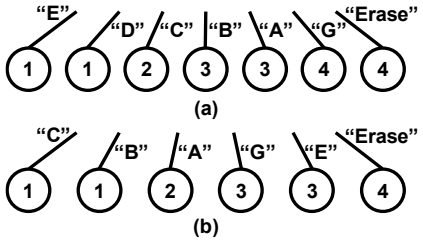


図 6 7 元、6 元ハフマン符号による提案手法[1]

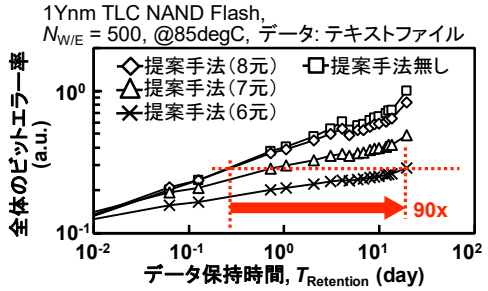


図 7 ビットエラー率測定結果[1]

4. 実測結果

テキストファイルを変調し実測したビットエラー率を図 7 に示す。TLC NAND 型フラッシュメモリの信頼性はデータ保持時間とともにビットエラーが増えていくが、提案の 8 元、7 元、6 元のハフマン符号を用いることで、許容可能なデータ保持時間はそれぞれ 1.7 倍、11 倍、90 倍に延ばすことができた。また、その時の圧縮率は、それぞれ 0.605、0.642、0.685 であり、7 元、6 元ハフマン符号の圧縮率が悪化した。これは、ステートを 1 つ又は 2 つ取り除いたことによるオーバーヘッドである。

5. 結論

TLC NAND 型フラッシュメモリの高信頼化のためにハフマン符号を応用した手法によって、最大 90 倍のデータ保持時間を許容できた。また、同時にデータ圧縮も実現し、その圧縮率は従来の 2 元ハフマン符号と同等に近い結果が得られた。

謝 辞

本研究の一部は JST、CREST の支援（グラント番号 JPMJCR1532）を受けたものである。

参 考 文 献

[1] Y. Deguchi et al., “Flash Reliability Boost Huffman Coding (FRBH): Co-Optimization of Data Compression and V_{TH} Distribution Modulation to Enhance Data-Retention Time by over 2900x,” *Symp. VLSI Tech. Dig. Tech. Papers*, pp. T206-207, Kyoto, Japan, June 2017.

[2] D. A. Huffman, “A Method for the Construction of Minimum-Redundancy Codes,” *Proc. I.R.E*, pp. 1098-1102, Sep. 1952.

[3] S. Tanakamaru et al., “Application-Aware Solid-State Drives (SSDs) with Adaptive Coding,” *Symp. VLSI Circ. Dig. Tech. Papers*, pp. 126-127, Honolulu, USA, June 2014.