CC-004

連想メモリベース超並列 SIMD 型演算コアによるリードソロモン符号の並列符号化処理について

Implementation of parellel Reed-Solomon Encoding with content addressable memory-based massive-parallel SIMD matrix processing core

濱野甫 * 蔭山享佑 † 小出哲士 ‡ 熊木武志 *
Hajime Hamano Kyosuke Kageyama Tetsushi Koide Takeshi Kumaki

1. はじめに

近年，スマートフォンを始めとする組込み機器の普及に伴い，それらに搭載されるハードウェア，ソフトウェアは共により高い性能が求められている．また，それら組込み機器は，通信技術の発展により，Wi-Fi や GPS 等，様々な通信技術を扱う必要がある．この際，近年では，誤り訂正符号の一種である，リードソロモン符号 [1] が多用されている．このアルゴリズムは組込み機器上で処理されている他のマルチメディアアプリケーションと同様に，繰り返し算術演算と，テーブルルックアップ処理によって構成されている．そして，他のマルチメディアアプリケーションと同様にテーブルルックアップ変換の高速化が困難であり，ボトルネックとなっている [2],[3].

そこで本論文では，組込み機器上でマルチメディアアプリケーションや機械学習処理を高速に実行するために開発している，並列処理プロセッサコア CAMX にリードソロモン符号の符号化アルゴリズムを実装する手法を提案する．CAMX はマルチメディアアプリケーションや機械学習処理を並列処理によって高速化するアーキテクチャであり，テーブルルックアップ変換の並列処理も可能である．この特徴を活かして，リードソロモン符号アルゴリズムの高速化を行い，実験結果を評価する．

2. 連想メモリベース超並列 SIMD 型演算コア

本章では，組込み機器への搭載を目的として，我々が開発しているプロセッサコアである，連想メモリベース超並列 SIMD 型演算コア（Content Addressable memory-based massive parallel SIMD matriX core: CAMX）について述べる．CAMX は組込み機器上でのマルチメディアアプリケーション処理や機械学習処理を高速に実行する事を目的としており，組込み機器向け CPU のアクセラレータとしての運用を想定している．

通常，マルチメディアアプリケーションや機械学習処理は繰り返し算術演算とテーブルルックアップ変換によって構成されている．繰り返し算術演算とはプログラムにおける for 文や loop 文の様に同一の演算を繰り返し何度も行う処理である．また，テーブルルックアップ変換は，入力データに対して予め用意されているテーブル（表）を参照して，対応するデータに非線形変換を施す処理である．繰り返し算術演算は，SIMD 構成を採ることにより並列処理が容易である [4]．一方，テーブルルックアップ変換は並列化による効率的な高速化が困難である [5].

そこで我々は，テーブルルックアップ変換を並列に処理する手段として連想メモリ [6] に着目した．連想メモリの例を図 1 に示す．連想メモリとは，CAM（Content Addressable Memory）とも呼ばれている，SRAM に比較器を付加したメモリである．一般的なメモリがアドレスを指定するとデータを出力するのに対して，連想メモリはデータを入力すると，対象データを格納しているアドレスを出力する．すなわち，メモリ内のデータの検索が可能である．この連想メモリを活用することで，テーブルルックアップ変換の並列処理が可能になる [7].

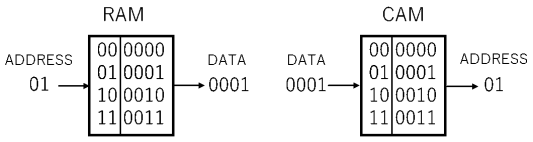

図 1: RAM と連想メモリ (CAM) の違い

CAMX は，繰り返し算術演算とテーブルルックアップ変換を，どちらも並列に処理するため，SIMD と連想メモリを融合させたアーキテクチャである [8]．図 2 に CAMX のブロック図を示す．2 つの 256 ビット ×1,024 エントリの連想メモリが 1,024 個の小型演算器 (Processing Element: PE) を挟み込む形で配置されている．ここで PE は 1 ビット演算器である．

各 PE は，Valid フラグ (図 2 の V) を持ち，これが High の場合は演算を実行，Low の場合は実行しないことで，並列処理に柔軟性を持たせている．また，この V フラグには連想メモリからの一致信号を格納することができる．その動作例を図 3 に示す．この場合，マスクデータの最下位ビットのみが 1 であるため，検索データの最下位ビットのみが検索される．よって，最下位ビットが 1 のデータを格納しているエントリのみが，一致信号を出力する．その一致信号を PE の Valid フラグに格納して演算を制御することができる．CAMX では，この機能を活用して並列処理に柔軟性を持たせている．また，一致信号が出力されているエントリにデータを一括に書き込むことで，テーブルルックアップ変換を並列に処理している．この提案コアによってマルチメディアアプリケーション及び，AI 処理の高速化が実現されている [9]–[11].

*立命館大学，Ritsumeikan Univ.
†近畿大学，Kindai Univ.
‡広島大学, Hiroshima Univ.

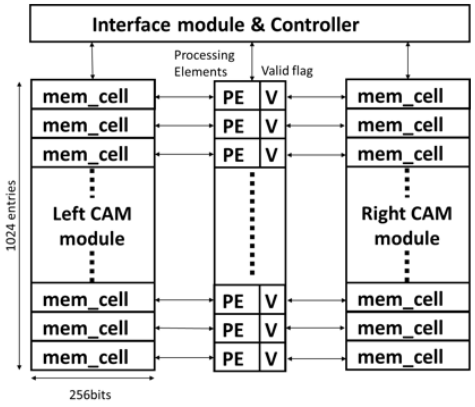

図 2: CAMX の全体構成

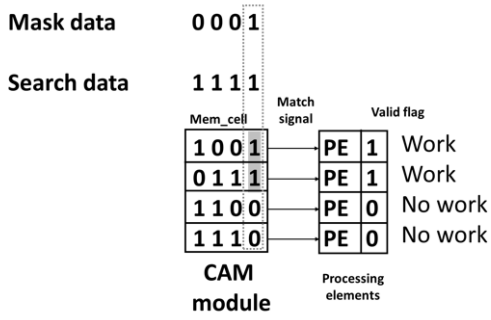

図 3: CAMX の動作例

3. リードソロモン符号

リードソロモン符号は誤り訂正符号の一種であり，連続する複数ビットに誤りが発生するバースト誤りなどに対して，強度を持つアルゴリズムである [12].

このアルゴリズムは，ガロア体と呼ばれる演算形態が用いられる [13]．ガロア体とは，四則演算が定義され，閉じている有限集合のことを指す．代数学分野における概念であり，一般には「有限体」と呼ばれており，主に計算機分野において用いられることが多い．ガロア体の演算では，拡大体の表を用いたテーブル変換と排他的論理和の演算を行うことで四則演算が処理できる．

リード・ソロモン符号は，r ビットの集まりを 1 シンボルとして，N 個のシンボルによる r × N ビットを符号語とし，K 個のシンボルを実際に送信する情報とすると，残りの N − K 個のシンボルが符号化により生成される．また，$t = (N - K)/2$ としたとき，この符号は t 個までのシンボルが訂正可能である．

この符号では r ビットの各シンボルを係数とした (N-1) 次の多項式の形として処理を行う．4 つのシンボル (A，B，C，D) が与えられた場合，これらを情報多項式 $I(x)$ として，式（1）のように表す．

$$I(x) = Ax^3 + Bx^2 + Cx + D \tag{1}$$

符号化では，生成された情報多項式 $I(x)$ に加えて，式（2）で表される生成多項式を用意する．また，この式の b には任意の整数を入れる．

$$G(x) = \prod_{i=b}^{2t-1+b} (x - \alpha^i) \tag{2}$$

情報多項式と生成多項式を用いて式（3）, 及び式（4）の演算を行う．

$$C(x) = x^{N-K} \times I(x) + P(x) \tag{3}$$

$$P(x) = x^{N-K} \times I(x)\ mod\ G(x) \tag{4}$$

そして，演算で生成された $C(x)$ が送信される符号となる．

4. 連想メモリベース超並列 SIMD 型演算コアによるリードソロモン符号の実装

本論文における実験では，リードソロモン符号で用いるパラメータを表 1 のように決定した．

表 1: 実験に用いるリードソロモン符号化のパラメータ

1 シンボルのビット数 (r)	8 ビット
送信シンボル個数 (N)	8 個
情報シンボル個数 (K)	4 個
訂正可能シンボル数 (t)	2 個

4 つの情報シンボルに対して符号化を行い，4 つの符号シンボルを生成し，情報シンボルの末尾に挿入することで，送信情報となる．この条件下では，符号長に対して 25% の訂正が可能である．また，CAMX では，一つの符号化を 1 エントリ内で処理するため，1,024 回分の符号化が一括に行える．

4.1. テーブルルックアップ変換の並列処理

CAMX では，ガロア体演算の拡大表に基づく，べき表現とベクトル表現のテーブルルックアップ変換を並列処理で実行する（図 4）．べき表現からベクトル表現に変換する際は，256 通りある，8 ビットのべき表現を検索し，対応するベクトル表現を一括に書き込むことで並列に変換処理が実行される．ベクトル表現をべき表現に変換する際も同様である．

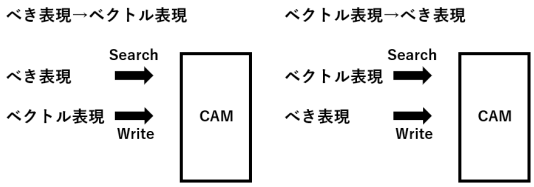

図 4: CAMX 上でのテーブルルックアップ変換の並列処理

4.2. 符号化の並列処理

今回，符号化で利用する生成多項式は，$b = 0$ として，式 (5) のような多項式となる．

$$G(x) = \prod_{i=0}^{3}(x - \alpha^i) = x^4 + \alpha^{75}x^3 + \alpha^{249}x^2 + \alpha^{78}x + \alpha^6 \quad (5)$$

符号化では，入力データからなる多項式 $I(x)$ を，上記で生成した生成多項式 $G(x)$ で除算し，その余りを誤り訂正符号として $I(x)$ の後に付加する．CAMX における，除算の処理を図 5 に示す．

5. 実装結果と評価

本章では，リード・ソロモン符号の符号化アルゴリズムを CAMX のシミュレータ上で実行した結果と，他プロセッサとの実行時間を比較して評価を行う．

5.1. 比較プロセッサの実験環境

組込み機器に多く搭載されている Arm コアを用いて，処理速度の比較検証を行った．Arm コアに内蔵されている，SIMD 演算を行う NEON [14] を用いたリードソロモン符号化の並列処理を行い，処理時間を測定した．実機は，Cortex-A72 を搭載する Raspberry Pi 4 を利用した．アルゴリズムは CAMX による処理と同様の動作を行う，C 言語のプログラムを用い，Raspberry Pi 4 の動作周波数は 1.5 GHz である．本論文では動作比較の対象として，Raspberry Pi 4 の CPU のみで動作させたもの「NONEON」と NEON を用いて処理させたもの「NEON」の 2 つの条件で比較を行った．比較データとして，プログラムを一度処理した処理時間に加えて，現在の CAMX の並列度に合わせて処理を 1,024 回繰り返した処理時間を取得した．また，各実行結果はそれぞれの処理を 10,000 回ループした処理時間の平均をとって算出している．表 2 に得られた処理時間を示す．

表 2: Raspberry Pi 4 の実行時間

	1,024 回処理 符号化 [ms]	1 回処理 符号化 [ms]
NEON	94.3650	0.1829
NONEON	373.6660	0.4177

5.2. 実行結果

リードソロモン符号の符号化処理を CAMX に実装した結果，全てのエントリにて，入力データが正しく符号化されていることが確認できた．また，CAMX を用いて符号化された送信情報に誤りを加えても正しく復号することができた．

5.3. リードソロモン符号の評価

CAMX の動作時間を算出するための実験環境として，Xillinx 社の VIVADO ver.2020.2 を用いて動作シミュレーションを行い，各処理のクロックサイクル数を出力した．得られたクロックサイクル数を表 3 に示す．この表のクロックサイクル数より実行時間を算出し，比較を行った．また本実験における CAMX の動作は，シミュレーション上での結果によるものであるため，動作周波数を仮定し設定したもので実行時間を算出している．

表 3: 符号化のクロックサイクル数

処理	サイクル数
符号化	399,053

CAMX による処理時間と表 2 との関係を示すグラフを図 6 に示す．このグラフから，CAMX は Raspberry Pi 4 の動作周波数と比べて符号化では約 0.28 倍の動作周波数でも同等の時間で処理が可能である．このことから，現在の CAMX の並列度 1,024 並列のリードソロモン符号の処理において，処理時間の面で CAMX は

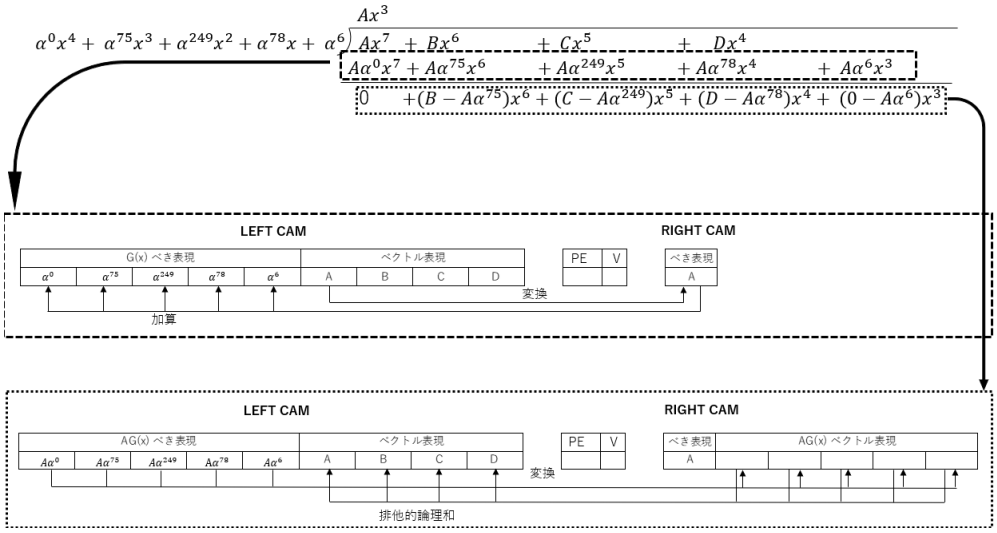

図 5: CAMX における除算処理

NEON を搭載した Raspberry Pi 4 よりも高速な処理が可能である事が分かる．

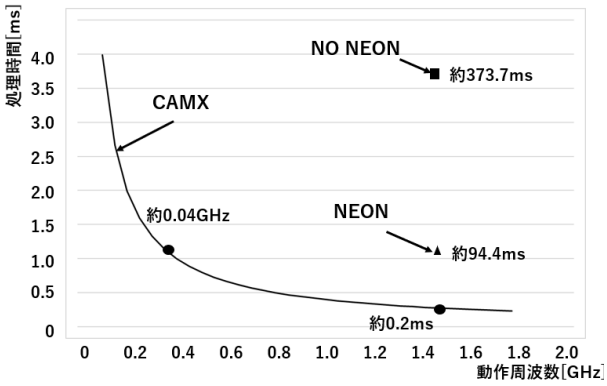

図 6: 符号化処理による CAMX の実行時間と Raspberry Pi 4 の実行時間

6. おわりに

本論文では，誤り訂正符号として広く使用されている，リードソロモン符号アルゴリズムを組込み機器向けプロセッサである CAMX に実装し，並列処理を行った．課題となっていたテーブルルックアップ変換を並列化する手法を示し高速化を実現した．また，他の組込み機器向けプロセッサとの比較においても，その優位性を示せた．同一周波数で CAMX を動作させ，比較プロセッサの処理時間と比べた場合，約 0.28%の処理時間で同一のアルゴリズムが処理できる．これにより，CAMX によるリードソロモン符号実装の有効性が示された．

参考文献

[1] W. A. Geisel, “Tutorial on reed-solomon error correction coding,” vol. 102162, 1990.

[2] M. Chi, D. He, and J. Liu, “Fast software-based table-less algorithm for crc generation,” 2018 21st International Symposium on Wireless Personal Multimedia Communications (WPMC), pp. 544–549, 2018.

[3] Y. Pan, N. Ge, and Z. Dong, “Crc look-up table optimization for single-bit error correction,” Tsinghua Science and Technology, vol. 12, no. 5, pp. 620–623, 2007.

[4] J. Wang, A. Karlsson, J. Sohl, and D. Liu, “Convolutional decoding on deep-pipelined simd processor with flexible parallel memory,” 2012 15th Euromicro Conference on Digital System Design, pp. 529–532, 2012.

[5] 若林俊一， 菅谷至寛， 阿曽弘具, “時間的・空間的並列処理による算術符号化・復号の高速アーキテクチャの提案,” 第 67 回全国大会講演論文集, vol. 2005, pp. 163–164, mar 2005.

[6] V. R. Datti and P. Sridevi, “Performance evaluation of content addressable memories,” 2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), pp. 596–598, 2018.

[7] J.-L. Lai, K.-H. Liao, Y.-T. Lai, and R.-J. Chen, “Design carom module used in aes structure for sub-byte and inv-sub-byte transformation,” 2013 International Symposium on Biometrics and Security Technologies, pp. 198–202, 2013.

[8] 熊木武志, “機能メモリベース超並列 SIMD 型演算コアの提案,” 第 15 回情報科学技術フォーラム FIT, vol. 1, no. C-001, pp. 233–234, Sept. 2016.

[9] K. Kageyama, A. Hamai, K. Watanabe, T. Koide, and T. Kumaki, “Floating-point arithmetic of content addressable memory-based massive-parallel SIMD matrix core,” Workshop on Nonlinear Circuits and Signal Processing, 2021.

[10] K. Kageyama, “Multiplication of baugh-wooley arithmetic processing by content addressable memory-based massive-parallel SIMD matrix core,” The International Symposium on Biomedical Engineering, 2021.

[11] H. Hamano, S. Arai, A. Hamai, K. Kageyama, K. Xiangbo, T. Koide, and T. Kumaki, “Implementation of self-organizing map with content addressable memory-based massive-parallel simd matrix processing core,” 2022 5th World Symposium on Communication Engineering (WSCE), pp. 100–104, 2022.

[12] I. Reed and G. Solomon, “Polynomial codes over certain finite fields,” Journal of the Society for Industrial and Applied Mathematics, vol. 8, no. 2, pp. 300–304, 1960.

[13] F. Geche, O. Mulesa, V. Voloshchuk, and A. Batyuk, “Generalized logical neural functions over the galois field and their properties,” 2019 IEEE 14th International Conference on Computer Sciences and Information Technologies (CSIT), vol. 1, pp. 21–24, 2019.

[14] https://www.arm.com/ja/technologies/neon.