様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

令和　４ 年　 ６ 月 １６ 日現在

機関番号：３４３１５
研究種目：基盤研究(C)（一般）
研究期間：2019～2021
課題番号：１９Ｋ０４４６１

研究課題名（和文）1000並列を超えるデータ処理を実現する連想メモリベース演算コアの開発

研究課題名（英文）Development of content addressable memory-based massive parallel SIMD matrix processing core

研究代表者

熊木　武志（Takeshi, Kumaki）

立命館大学・理工学部・教授

研究者番号：６０４５２５９６

交付決定額（研究期間全体）：（直接経費）　　3,400,000円

研究成果の概要（和文）：モバイル機器に搭載する超並列処理プロセッサを開発した．構成，LSIに実装するための回路データから，プログラミングのための命令コマンドを作成した．本プロセッサは，連想メモリと呼ばれる検索処理に特化した回路をベースにしたものであり，これに小規模な演算器を数千個配置できる．本プロセッサは連想メモリの検索機能を活かしたテーブル変換処理と，演算器による論理・数値演算の両立が可能である．これを活用して，乗算，浮動小数点演算，ブロック暗号処理（AES，Present），機械学習（自己組織化マップ），及び画像処理（モルフォロジカル変換)を実装し，Armコア等と比較して性能面での優位性を確認した．

研究成果の学術的意義や社会的意義
近年，スマートフォンに代表されるモバイル機器の重要性が高まっている．例えば，店頭での支払いだけでなく，コロナ禍における陰性証明等が挙げられ，もはや必需品と言える．しかしながら，5Gの普及による通信の速度向上はよく耳にするが，内部に組み込まれている回路の性能向上が必要であることはあまり報道されていない．これは，様々なアルゴリズムの処理速度を向上させるのに直接かかわるものであり，かつ最も重要なハードウェアである．以上の事から，我々はモバイル機器上で様々なアプリを超並列に処理可能な連想メモリベースのプロセッサを開発し，様々な処理を行い，既存のプロセッサと比較することでその効果を確認できた．

研究成果の概要（英文）：Massive parallel processor is developed for mobile devices. Architecture, circuit data, basic instruction command are implemented. This processor is based on content addressable memory and has over thousand processing elements. Then, table lookup operation and logic or numerical calculation are executed on this processor. For verifying processing capability, multiplication, floating point operation, block cipher (AES, Present), machine learning (SOM) and image processing (Morphological method) are implemented on this processor. We can obtain effective results to compare with Arm core.

研究分野：電子情報工学

キーワード：モバイル機器　並列処理　連想メモリ　マルチメディアデータ　スマートフォン　LSI

１．研究開始当初の背景

現在，モバイル機器には世界的にシェアが高い ARM 社が開発したプロセッサコアが用いられており，この内部に使用されている並列演算コアは NEON と呼ばれている．今後はエッジコンピューティングと呼ばれる，クラウドで機械学習を始めとした様々な処理を行うのではなく，センシングデータ取得時やネットワークの端末で処理を行い，クラウドでは処理後のデータを分析するという思想が中心となる．この様な流れに向かっていく中で組込み機器向けのプロセッサが持つ並列度は，前述した NEON でたかだか 32 程度であり，画像処理や深層学習の効率的な実行は困難である．それにもかかわらず，この分野は技術進展が進んでいない．研究代表者は，ルネサステクノロジと組込み機器向け超並列マルチメディア処理演算コア（超並列演算コア)を実用化した経験がある．この超並列演算コアは，四則・論理演算を並列に処理が可能であるものの，様々なアルゴリズムを効率よく実装するためには，図 2 に示す通り，マルチメディアアプリケーションの特性を考慮し，繰り返し演算処理とテーブル変換処理の両立が必要不可欠となる．超並列マルチメディア処理演算コアは構成上テーブル変換処理が難しく性能の限界が判明していた．そこで本研究では，テーブル変換処理は連想処理と関連があることに着目し，連想メモリをベースとした新しい超並列演算コアを実現することとした．

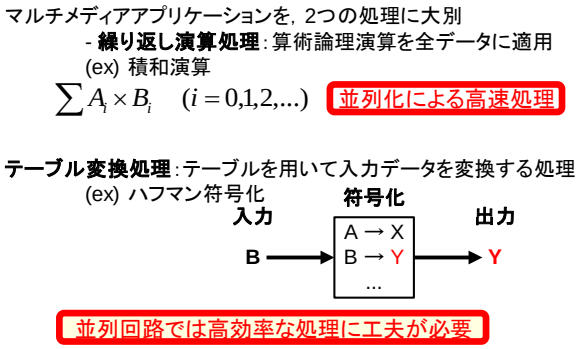

図 1. 並列化とマルチメディアアプリケーション

２．研究の目的

本研究の目的とは，1,000 を超える算術論理演算とテーブル変換をモバイル機器上で両立し，機械学習を始めとするマルチメディアアプリケーションを高速に実行することである．

連想メモリの起源は古く，1956 年に発表されたクライオトロン・カタログメモリと呼ばれる回路が源であり，現在では，その高速な検索処理（図 2)を活かしてネットワークルータにおけるパケットルーティングに使用されている．

一方，近年の AI ブームは，深層学習のアルゴリズム的な改良から，実社会への効率的な LSI 実装方法へ人々の関心が移り変わってきており，ここに連想メモリを活かす事ができると考えている．更に算術演算も同時に効率よくこなす必要があることから，連想メモリと超並列演算コアの融合は，両方の研究で論文のみならず，特許や受賞暦があり，精通した研究代表者でなければ開発は困難である．

超並列演算コアは，2 ビット単位でデータを逐次に読み出しつつ，1,000 を超える並列度を実現するため，SRAM の向きを水平方向にすることでビット幅をそのまま並列度としている（図 3，4)．しかしながらこの構成は，データの格納に前処理が必要となり，データ間のやり取りも困難であるため，テーブル変換処理に問題が生じていた．また，LSI を設計する際にもレイアウトが特殊な形状となる．連想メモリは図 5 に示す通り，一般的なメモリと同じく垂直方向にデータを読み書きしながら，検索処理にマスクと呼ばれるドントケア機能を利用することで，水平方向にデータを読み出すことも

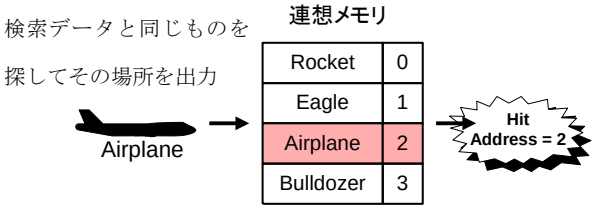

図 2. 連想メモリの処理イメージ

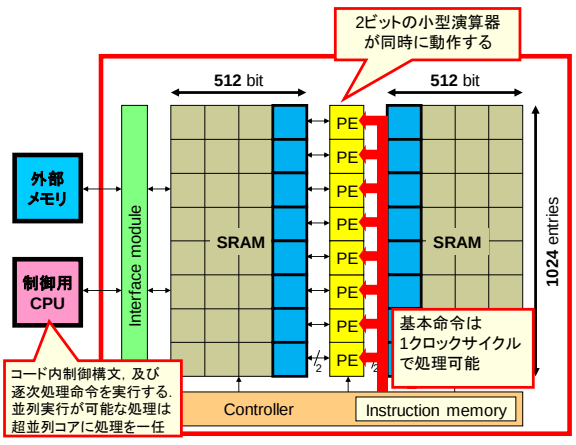

図 3. 超並列マルチメディア処理演算コア

きる．これはアドレスで表現するワード数分の逐次並列処理ができることを示している．すなわち，前処理やレイアウトの変更も不要であり，基本機能である連想処理でテーブル変換も高速に行うことができる．機械学習を始めとする，マルチメディアアプリケーションは算術論理演算とテーブル変換処理から構成されることが多いために，これらの処理を十分にカバーすることが可能となる．

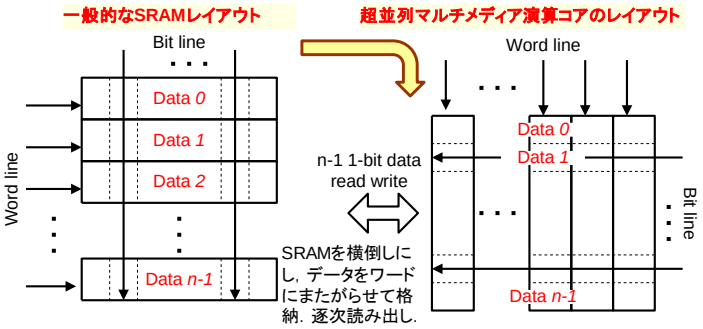

図 4. 超並列を実現するための工夫

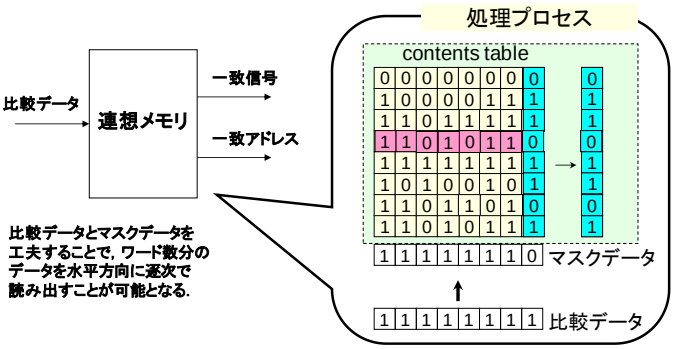

図 5. 連想メモリを用いた並列逐次読み出し

3．研究の方法

開発目標である連想メモリベース超並列演算コアのブロック図を図 6 に示す．プロトタイプは，1,024 個の 1 ビット演算器を垂直に配置し，512 ビット，1,024 ワードの連想メモリを 2 つ用意した上で演算器群を両側から挟み込んでいる．この構成はデータを左，右，左，右・・・と交互かつ水平方向に読み出す事ができ，パイプラインにデータを連続して 1,024 並列に処理する事が可能である．また，アルゴリズム中にテーブル変換処理がある場合においては，図 2 に示した検索処理を用いて垂直方向の連想処理を行う．

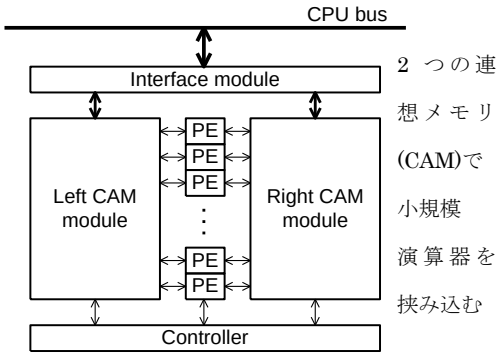

図 6. 連想メモリベース超並列演算コア

始めに，提案コアをハードウェア記述言語で設計し，その動作を波形動作シミュレーションで確認する．その後，モバイル機器で用いられることの多い画像処理（JPEG）と暗号化処理（AES）を実装する．これらは，算術論理演算と共に，ハフマン符号化と SubBytes と呼ばれるテーブル変換処理を含んでいるためベンチマークとして適しているからである．

次に，提案コアの動作をシミュレーションで確認するだけでなく，ハードウェアとして確実に動作するかを確認するために，図 7 に示す，FPGA を搭載した評価ボードを用いて実機検証を行う．このボードはモバイル機器向けプロセッサの実機検証のために，研究代表者が以前特注で開発したものである．このボードは FPGA の交換が可能であり，我々が開発した様々な回路に CPU を介して性能を評価することが可能である．開発して数年経過しているため，搭載 FPGA の規模が小規模であり，十分な検証は難しいが，まずは連想メモリのサイズをコンパクトにして動作検証を行う．

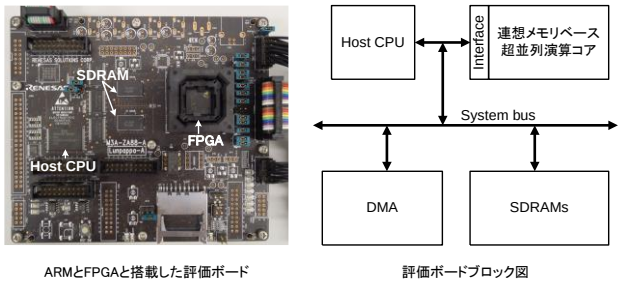

図 7. ARM と FPGA と搭載した評価ボード

以上に加えて，機械学習の処理をシミュレーションにて動作検証する．これらは，モバイル機器におけるデータの読み取りとその後の AI 処理を想定したアプリケーションの組み合わせとなっている．

４．研究成果

実施を行った項目ごとにその成果について述べる．

・コマンドによる基本動作の確認

連想メモリベース演算コアをハードウェア記述言語 Verilog-HDL で作成し，更に専用の命令群を定義した．この演算コアの基本構成は 128 ビット 1024 ワードの連想メモリを 2 つ，1 ビットの演算器を 1024 個用意している．それを用いて，基本的な処理である四則演算，論理演算，テーブル変換処理の動作を検証した．その方法としては Verilog-HDL を用いた論理シミュレーションであり演算や動作の結果に矛盾がないことを確認した．付録にコマンド群の例を示す．

・実機により動作検証

連想メモリベース超並列 SIMD 型演算コアの FPGA による動作を検証した．ターゲットの FPGA は，Xillinx 社の Spartan6 ファミリに含まれる XC6SLX150 である．開発ツールには Xilinx 社の ISE14.7，ISim14.7，及び IMPACT を用いた．コアの実装に関しては FPGA の容量制限もあり 64 とした．また，動作検証を行うにあたって，基本命令である加算を並列に実行した．図 8 に提案コアによって 23 + 42 の加算結果を 10 進数で 7 セグメント LED に出力した例を示す．

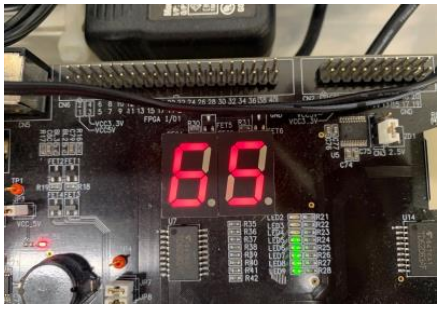

図 8．超並列連想メモリコアによる加算結果

・乗算処理

連想メモリベース超並列演算コアに適した演算方法を調査するために，ビットシリアル乗算，検索・加算繰り返し乗算，及び Baugh-Wooley 乗算の実装を行った．その結果，データ長が 15 ビットを境に，検索・加算繰り返し乗算，そして Baugh-Wooley 乗算が高速であることが分かった．

・暗号化処理

暗号化処理は四則・論理演算とテーブル変換を含んでいるものであり，特にブロック暗号は S-box と呼ばれる変換処理を含んでいることが知られている．本研究では AES と軽量ブロック暗号である PRESENT の実装を行った．既存の暗号処理回路は複数のテーブルを用意して並列化を行っていたが，連想メモリベース超並列演算コアは先に述べた基本コマンドを組合わせて，超並列に繰り返し論理演算を行いつつ，テーブル変換を並列に行えることができる．

実装した結果，1,024 の平文が並列に処理され正確に暗号化されていることが確認できた．この得られた結果と RaspberryPi4 に搭載されている Arm コアとで比較を実施した．その結果，Arm コアによる最適化を行った処理と AES と比較して約 2.5 倍の処理速度向上が確認できた．また，PRESENT では同条件で約 6.87 倍の処理速度向上が図れた．

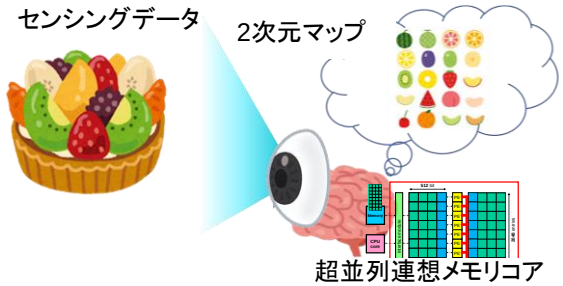

図 9．超並列連想メモリコアによる大脳視覚野処理

・機械学習

自己組織化マップ (Self-Organizing Map: SOM)の実装を行った．本アルゴリズムは並列化に適しており，イメージデータの分類を中心に利用する事ができる (図 9)．また，入力対象の特徴を計算するためにモルフォロジカルウェーブレット変換と呼ばれる方法を用いた．このアルゴリズムは加算，減算を最大値・最小値検索を利用する事で画像の周波数分解を可能とするものであり並列処理に向いている．図 10 に結果を示す，色の識別を行ったところ，Arm コア NEON に対して約 2.5 倍の処理速度向上を確認している．

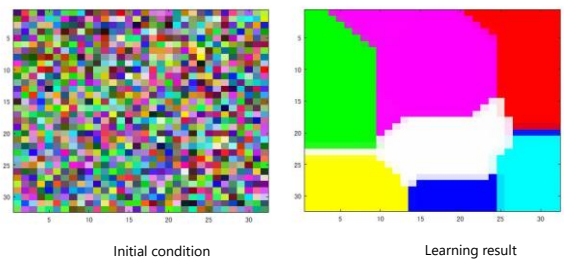

図 10．自己組織化マップのテスト実装

（付録）

/**==========CAMX を動作させるコマンド群==========**/

CAMX を動作させるためには，CAMXLIB コマンドポートに 32 ビットのデータを入力する．
CAMXLIB のビット区分は以下の通り．

CAMXLIB = {翼/PE 選択，動作命令，C ビット，B ポインタ，A ポインタ}
= {2'bxxx, 6'bxxxx_xxxx, 8'bxxx_xxxx, 8'bxxx_xxxx, 8'bxxx_xxxx}

/**-----翼/PE 選択----------**/
LEFT_WING: 左翼を選択
RIGHT_WING: 右翼を選択
BOTH_WING: 両翼選択 //###未実装###
PE_EXE: 演算器を選択（翼は動作させない）

/**-----動作命令（CAM に対する動作コマンド）----------**/

・CAMX_CNTRL_CLEAR: CAMX コントローラのレジスタクリア

・CAMX_DATA_WRITE: 左/右翼の CAM 内ワードにデータを書き込む（同時にアドレスとデータを入力）．A，B ポインタ，及び C ビットは 0 にしておく．
（例）右翼アドレス 10'b00_0000_0011 に，データ 256'h0000_0000_0000_0000_0000_0000_0000_0000_0000_ffff_0000_ffff_0000_ffff_0000_ffff を書き込む

```
#(STEP)        ADDRESS = 10'b00_0000_0011;
               DIN = 256'h0000_0000_0000_0000_0000_0000_0000_0000_0000_ffff_0000_ffff_0000_ffff_0000_ffff;
               CAMXLIB = {RIGHT_WING, CAMX_DATA_WRITE, 8'b0000_0000, 8'b0000_0000, 8'b0000_0000};
```

・CAMX_DATA_READ: 左/右翼の CAM 内ワードからデータを読み出す（同時にアドレスを入力）．A，B ポインタ，及び C ビットは 0 にしておく．
DOUT_VALID が H の時の DOUT が読み出されたデータとなる．
（例）左翼アドレス 10'b00_0000_0011 から，データを読み出す．

```
#(STEP)        ADDRESS = 10'b00_0000_0011;
               CAMXLIB = {LEFT_WING, CAMX_DATA_READ, 8'b0000_0000, 8'b0000_0000, 8'b0000_0000};
```

・CAMX_UNMASK_SEARCH: 左/右翼の CAM に対し，マスクを用いない検索処理を行う（同時に検索データを入力）．A，B ポインタ，及び C ビットは 0 にしておく．
また，検索結果は外部に出力されると共に，PE 内のレジスタに格納される．
ただし，検索結果は PE 内のレジスタにも保存される．
（例）検索データ 256'h0000_0000_0000_0000_0000_0000_0000_0000_0000_ffff_0000_ffff_0000_ffff_0000_0000 を右翼に入力し，全データの一致検索処理．

```
#(STEP)        SEARCH_DIN = 256'h0000_0000_0000_0000_0000_0000_0000_0000_0000_ffff_0000_ffff_0000_ffff_0000_0000;
               CAMXLIB = {RIGHT_WING, CAMX_UNMASK_SEARCH, 8'b0000_0000, 8'b0000_0000, 8'b0000_0000};
```

・CAMX_MASK_SEARCH: 左/右翼の CAM に対し，マスクを用いた検索処理を行う（同時に検索データ，マスクを入力）．A，B ポインタ，及び C ビットは 0 にしておく．マスクは 0 で掛ける．また，検索結果は外部に出力されると共に，PE 内のレジスタに格納される．
ただし，検索結果は PE 内のレジスタにも保存される．
（例）検索データ 256'h0000_0000_0000_0000_0000_0000_0000_0000_0000_ffff_0000_ffff_0000_ffff_0000_0000 を右翼に入力し，マスクデータ 256'h0000_0000_0000_0000_0000_0000_0000_0000_0000_0000_0000_0000_0000_0000_ffff_ffff を用いて一致検索処理．

```
#(STEP)        SEARCH_DIN = 256'h0000_0000_0000_0000_0000_0000_0000_0000_0000_ffff_0000_ffff_0000_ffff_0000_0000;
               MASK_DIN = 256'h0000_0000_0000_0000_0000_0000_0000_0000_0000_0000_0000_0000_0000_0000_ffff_ffff;
               CAMXLIB = {RIGHT_WING, CAMX_UNMASK_SEARCH, 8'b0000_0000, 8'b0000_0000, 8'b0000_0000};
```

/**-----動作命令（PE も含めた動作コマンド）----------**/

・CAMX_DATA_REG: CAM からの読み出し値を PE 内のオペレーションレジスタに書き込む．A ポインタを使用，B ポインタ，及び C ビットは 0 にしておく．
（例）左翼 8'b0000_1100 の位置にあるビットを PE 内のオペレーションレジスタに格納．

```
#(STEP)        CAMXLIB = {LEFT_WING, CAMX_DATA_REG, 8'b0000_0000, 8'b0000_0000, 8'b0000_1100};
```

・CAMX_REG_DATA: PE 内のオペレーションレジスタの値を CAM 左/右翼の A ポインタへ格納する．A ポインタを使用，B ポインタ，及び C ビットは 0 にしておく．
（例）PE 内のオペレーションレジスタに格納されているデータを，右翼 8'b0000_0001 へ書き込み．

```
#(STEP)        CAMXLIB = {RIGHT_WING, CAMX_REG_DATA, 8'b0000_0000, 8'b0000_0000, 8'b0000_0001};
```

・CAMX_DATA_AND: CAM からの読み出し値を PE 内で AND する．始めに左/右翼から A ポインタの値に従って読み出し，次に右/左翼から B ポインタの値に従って読み出し，C ビットの演算を行った後に，左/右翼の A ポインタから書き込む．
（例）左翼 8'b0100_0000 のデータを読み出しオペレーションレジスタへ格納，次に右翼 8'b0000_0000 のデータを読み出し，8'b0000_0111 ビットの AND 演算を行い，左翼 8'b0100_0000 から結果を格納する．

```
#(STEP)        CAMXLIB = {LEFT_WING, CAMX_DATA_AND, 8'b0000_0111, 8'b0000_0000, 8'b0100_0000};      //C ポインタビット幅は+1 で考える
```

・CAMX_DATA_OR: CAM からの読み出し値を PE 内で OR する．始めに左/右翼から A ポインタの値に従って読み出し，次に右/左翼から B ポインタの値に従って読み出し，C ビットの演算を行った後に，左/右翼の A ポインタから書き込む．
（例）右翼 8'b0000_0000 のデータを読み出しオペレーションレジスタへ格納，次に右翼 8'b0000_0000 のデータを読み出し，8'b0111_1111 ビットの OR 演算を行い，左翼 8'b0000_0000 から結果を格納する．

```
#(STEP)        CAMXLIB = {RIGHT_WING, CAMX_DATA_OR, 8'b0111_1111, 8'b0000_0000, 8'b0000_0000};      //C ポインタビット幅は+1 で考える
```

・CAMX_DATA_XOR: CAM からの読み出し値を PE 内で XOR する．始めに左/右翼から読み出し，次に右/左翼から読み出し，演算を行った後に，左/右翼に書き込む．
（例）左翼 8'b0000_0011 のデータを読み出しオペレーションレジスタへ格納，次に右翼 8'b0000_0010 のデータを読み出し，8'b0111_0000 ビットの XOR 演算を行い，左翼 8'b0000_0011 から結果を格納する．

```
#(STEP)        CAMXLIB = {LEFT_WING, CAMX_DATA_XOR, 8'b0111_0000, 8'b0000_0010, 8'b0000_0011};      //C ポインタビット幅は+1 で考える
```

・CAX_DATA_NOT: 左/右翼 CAM のデータを A ポインタで示した位置から，C ビット分 NOT する．A ポインタ，及び C ビットを使用，B ポインタは 0 にしておく．
（例）左翼 8'b0110_0000 から，8'b0000_1111 分のデータを反転する．

```
#(STEP)        CAMXLIB = {LEFT_WING, CAMX_DATA_NOT, 8'b0000_1111, 8'b0000_0000, 8'b0110_0000};      //左（右)翼のデータを否定，A: 読み込みポインタ，B: 不使用
```

５．主な発表論文等

〔雑誌論文〕　計0件

〔学会発表〕　計11件（うち招待講演　0件／うち国際学会　4件）

１．発表者名 Kyosuke Kageyama, Kensuke Watanabe, Akimitsu Hamai, Takeshi Kumaki and Tetsushi Koide
２．発表標題 "Acceleration of arithmetic processing with CAM-based massive-parallel SIMD matrix core
３．学会等名 IEEE International MidWest Symposium on Circuits And Systems (MWSCAS)（国際学会）
４．発表年 2020年

１．発表者名 Kyosuke Kageyama, Akimitsu Hamai, Kensuke Watanabe, Tetsushi Koide and Takeshi Kumaki
２．発表標題 Floating-point arithmetic of content addressable memory-based massive-parallel SIMD matrix core
３．学会等名 RISP International workshop on Nonlinear Circuit, computer and Signal Processing (NCSP)（国際学会）
４．発表年 2021年

１．発表者名 Kensuke Watanabe, Kyosuke Kageyama, Akira Sekino, Akimitsu Hamai, Takeshi Kumaki and Tetsushi Koide
２．発表標題 Basic operation verification of content addressable memory-based massive-parallel SIMD matrix core for multimedia applications
３．学会等名 International symposium on biomedical engineering (ISBE)（国際学会）
４．発表年 2019年

１．発表者名 Kyosuke Kageyama, Akira Sekino, Kensuke Watanabe, Akimitsu Hamai, Tetsushi Koide, and Takeshi Kumaki
２．発表標題 Proposal of content addressable memory-based massive-parallel SIMD matrix core
３．学会等名 RISP International workshop on Nonlinear Circuit, computer and Signal Processing (NCSP)（国際学会）
４．発表年 2020年

１．発表者名
渡辺健介，関野　輝，蔭山享佑，小出哲士，熊木武志

２．発表標題
連想メモリベース超並列SIMD型演算コアのシミュレーションによる動作検証について

３．学会等名
LSIとシステムのワークショップ2019

４．発表年
2019年

１．発表者名
蔭山享佑，関野　輝，渡辺健介，濱井彰光，熊木武志

２．発表標題
超並列マルチメディア処理CAMベースSIMD演算コア

３．学会等名
VDECデザイナーズフォーラム

４．発表年
2019年

１．発表者名
蔭山享佑，濱井彰光，荒井聡太，濱野　甫，孔　祥博，小出哲士，熊木武志

２．発表標題
連想メモリベース超並列SIMD型演算コアを用いた乗算手法の実装と評価

３．学会等名
第195回システムとLSIの設計技術研究発表会（新企画）

４．発表年
2021年

１．発表者名
濱野　甫，荒井聡太，濱井彰光，蔭山享佑，孔　祥博，小出哲士，熊木武志

２．発表標題
連想メモリベース超並列SIMD型演算コアのFPGAによる動作検証

３．学会等名
第195回システムとLSIの設計技術研究発表会（新企画）

４．発表年
2021年

１．発表者名
Sota Arai, Kyosuke Kageyama, Xiangbo Kong and Takeshi Kumaki

２．発表標題
Implementation and evaluation of block cipher algorithm with content addressable memory-based massive-parallel SIMD matrix core

３．学会等名
IEEE Grobal Conference on Consumer Electronics (GCCE)

４．発表年
2021年

１．発表者名
濱野　甫, 熊木武志

２．発表標題
一度に1000以上のデータ処理を可能にする連想演算回路の開発

３．学会等名
超異分野学会大阪大会2021

４．発表年
2021年

１．発表者名
Kyosuke Kageyama

２．発表標題
Multiplication of Baugh-wooley arithmetic processing by content addressable memory-based massive-parallel SIMD matrix core

３．学会等名
International symposium on biomedical engineering (ISBE)

４．発表年
2021年

〔図書〕　計0件

〔産業財産権〕

〔その他〕
マルチメディア集積回路システム研究室 - 熊木研究室
http://www.ritsumei.ac.jp/~kumaki/kumaki_hp/index.html

６．研究組織

	氏名 （ローマ字氏名） （研究者番号）	所属研究機関・部局・職 （機関番号）	備考

７．科研費を使用して開催した国際研究集会

〔国際研究集会〕　計0件

８．本研究に関連して実施した国際共同研究の実施状況

共同研究相手国	相手方研究機関