命令キャッシュ導入によるフラッシュメモリ搭載マイコンの低電力化

金　多厚[1]　平尾 岳志[1]　肥田　格[1]　浅井 哲也[1]　本村 真人[1]

概要：近年、様々な組み込み機器で使われるマイコンの低消費エネルギー化への要求が高まっている．多くのマイコンはフラッシュメモリに代表される不揮発性メモリをプログラムメモリとして搭載することでユーザープログラマブル化を実現しているが、一方ではこのプログラムメモリアクセスの際に大きな電力を消費してしまうという問題が存在する（低消費エネルギー化のために低電圧動作を狙う場合に、この問題は特に顕著となる）．本稿では、命令キャッシュを導入して不揮発性メモリアクセスを減らすことでこの問題の解決を目指す．通常のプロセッサの場合と違い、キャッシュミスの際のミスペナルティが許されないため、高性能化目的ではなく、消費電力削減のためのキャッシュ導入となる点が大きな特徴である．本稿では、そのアーキテクチャ上の工夫点と低電力化効果について報告する．

1. はじめに

近年、スマートソサイエティの実現を支える基盤技術としてセンサネットワーク技術が注目されている．センサネットワークが様々な応用分野で実現されるためには、小さいエネルギー源でも長時間駆動できるセンサノードが必要となり、センサノードを駆動するマイコンの低電力化を行う必要がある．

近年のマイコンは、ソフトウェアの書き換えが可能で、ソフトウェア開発が容易である不揮発性メモリをプログラムメモリとして搭載したマイコンが大きな割合を占めている．しかし、不揮発性メモリを搭載したマイコンの場合、不揮発性メモリ以外のロジック側は低電圧化が容易に出来る事に対し、不揮発性メモリは低電圧化が困難であり、不揮発性メモリの消費電力が占める割合も大きい [2]．本研究ではこの点に着目し、不揮発性メモリの中でも多く使われているフラッシュメモリを搭載したマイコンに命令キャッシュメモリを搭載するためのアーキテクチャ上の工夫を施し、その低電力化効果をシミュレーション評価することにした．

本研究で提案する命令キャッシュメモリは、低電力化が目的であり、ユーザー側から、キャッシュ導入が見えないようにすることが要求されている．このような要求を満たすためには、既存のキャッシュに存在していたミスペナルティを発生させない事が必要となる．

また本研究では、命令キャッシュ導入による低電力化効果を示すために、ルネサスエレクトロニクス社の 78K0R のアーキテクチャを参考にしたベースマイコンを RTL 実装し、その消費電力を見積もる事で電力評価を行った．ベースマイコンの RTL 実装は、ルネサスエレクトロニクス社の公開情報のみに基づいてアーキテクチャを推定しているため、78K0R のアーキテクチャと異なる可能性がある．

本論文の構成は以下の通りである．2 章でベースマイコンのアーキテクチャについて説明し、そのマイコンにキャッシュメモリを搭載するためのアーキテクチャの概要を説明する．3 章では、命令キャッシュ導入後のアーキテクチャの詳細な設計を説明する．4 章では、消費電力評価を行い、その結果を示す．5 章で総括する．

2. 命令キャッシュ導入アーキテクチャ概要

本章では、ベースマイコンのアーキテクチャに関して簡単に説明し、ベースマイコンにキャッシュメモリを搭載するためのアーキテクチャの概要を説明する．

2.1 ベースマイコンのアーキテクチャ

ベースマイコンのメモリは、78K0R のアーキテクチャと同様に、内部メモリとしてフラッシュメモリと SRAM を使用する [2]．フラッシュメモリは主に命令メモリとして、SRAM は主にデータメモリとして使用する．

ベースマイコンのアーキテクチャは、図 1 のベースマイコン部（Base Microcontroller）で示すように、3 段のパイ

[1] 北海道大学 大学院情報科学研究科
Graduate School of Information Science and Technology（IST）, Hokkaido University

プライン構造をしており、Instruction Fetch（IF）ステージ、Instruction Decode（ID）ステージ、Memory Access（MEM）ステージに分けられる [3]．IF ステージは、命令のフェッチを行うステージであり、プログラムカウンタ（PC） から、命令メモリのアドレスを指定し、命令メモリから命令をフェッチしてくる．ID ステージでは、IF ステージでフェッチしてきた命令をデコードし、制御信号を作る．また、データメモリのアドレスを生成する．最後の MEM ステージでは、実際の演算を行い、その演算結果をデータメモリや汎用レジスタに格納する動作をする [4]．

また、78K0R のアーキテクチャと同様に、CISC アーキテクチャを採用しており、1Byte〜5Byte の命令長を持つ命令を有する [4] ものとする．また、4Byte の命令キューを持っており [4][5]、フラッシュメモリから読み出した命令列を命令キューに格納し、命令キューに格納されているデータを用いて命令を実行する [4]．78K0R の場合、命令によって実行にかかるサイクル数が一定ではないが、命令長が 4Byte の命令であっても 1 サイクルで実行される命令も存在する [4]．したがって、フラッシュメモリのビット幅が 4Byte より小さいと 1 サイクルで命令キューに必要なデータを揃わせる事が出来なく、4Byte の命令は 1 サイクルで実行させる事ができなくなる．したがって、フラッシュメモリのビット幅は 4Byte とした．命令キューに 4Byte の命令列が揃っている場合、1〜3Byte の命令が実行されると、まだ、命令キューに実行できる命令列が存在するため、フラッシュメモリにアクセスする必要がない．また、実行に複数サイクルが掛かる命令を実行している間にも、フラッシュメモリにアクセスしない．したがって、フラッシュメモリに毎サイクルアクセスするのではなく、アクセスしないサイクルも存在する．

ベースマイコンの RTL 実装後は、78K0R のソフトウェア開発環境である Cubesuite+[6] を用い、EEMBC の Coremark[7]、78K0R アプリケーションノート [8] のアプリケーションのバイナリコードを生成し、シミュレーションの動作を確認することで RTL 検証を行った．

2.2 キャッシュの制約

本研究においてマイコンにキャッシュを導入する上での制約は、ユーザーにキャッシュが導入されたことを見えないようにすることである．つまり、キャッシュミス時のミスペナルティを発生させないことが必要となる．これは、既存のマイコンベースの組込みシステムで開発されたソフトウェアに影響がないようにすることや既存の開発方式でソフトウェア開発ができるようにすることが要求されているためである．また、実時間制御等の一部のマイコン応用では、プログラム実行時間をあらかじめ計算できることが重要であり、この場合はキャッシュミスにより実行時間が変動すること自体許容できない．

このようなことから、以下の方針でキャッシュを導入することにした．

(1) **フラッシュメモリ・命令キャッシュ並列アクセス方式**
ヒット判定を行う前にフラッシュメモリアクセスの信号と命令キャッシュキャッシュアクセスの信号が同時に用意されており、ミス時にフラッシュメモリから命令列を読み込んだ後に、命令キャッシュにその命令列を書き込む動作をする．キャッシュヒット時にはキャッシュから命令を読み込む方式である．

(2) **ダイレクトマップ方式**
キャッシュ導入により発生する遅延を短くする為に、ヒット判定回路が単純であり、高速にヒット判定ができるダイレクトマップ方式のキャッシュを導入する．

2.3 1 ライン 1 ワード命令キャッシュ

図 1 に、以上の方針に基づき命令キャッシュを導入したベースマイコンのブロック図を示す．ここで、フラッシュメモリから 1 サイクルで読み出す 1 ワードのビット幅が 32 ビットであることから、ミスペナルティを発生させない為に、命令キャッシュの 1 ラインのビット幅を 32 ビットにした．フラッシュメモリは、クロックに同期して入力データを読み取るクロック同期式のフラッシュメモリとして評価した．この方式におけるフラッシュメモリのアクセスタイミングは図 2 のように、プログラムカウンター（PC）でフラッシュメモリにアドレスを与えると、その次のサイクルの間にフラッシュメモリのデータを読み出す．命令キャッシュは、フラッシュメモリと同様にクロック同期式とし、SRAM として評価した．Tag メモリも命令キャッシュと同様にクロック同期式の SRAM として評価した．Tag メモリに関しては、レジスタで実装することも可能であるが、小型化に有利であり、低コストで実装できる SRAM を用いることにした．命令キャッシュ導入後のメモリアクセスタイミングは、図 3 に示すように、ヒットを判定後、キャッシュヒットの場合は、キャッシュ読み出し信号（Cache Read Enable）を命令キャッシュに与え、命令キャッシュから命令列を読み出す．キャッシュミスの場合は、フラッシュメモリ読み出し信号（Flash Read Enable）をフラッシュメモリに与え、フラッシュメモリから命令列を読み出す．

また、キャッシュミスの場合は、フラッシュメモリから 1 ワードのデータを読み出し、命令キューに格納するのと同時に読み出した命令列を命令キャッシュに書き込む動作をする．

2.4 命令キャッシュの 1 ライン 4 ワード化

前節では、1 つのラインに 1 つだけのワード（32bit）を格納する 1 ライン 1 ワードの命令キャッシュを説明した．しかし、命令データの空間的局所性をさらに活かすためには、命令キャッシュのラインサイズを複数ワードに拡張す

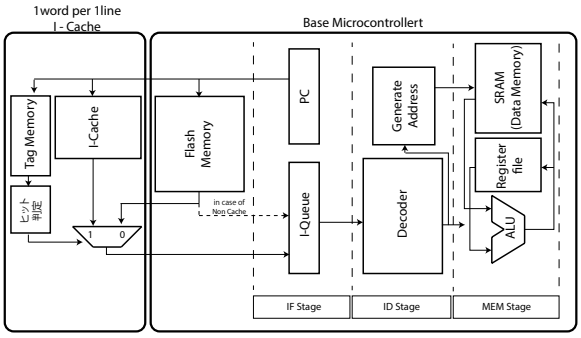


図 1 1 ライン 1 ワードの命令キャッシュを導入した 78K0R の概略ブロック図

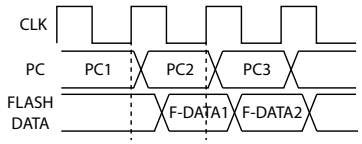


図 2 キャッシュ導入前のフラッシュメモリアクセスタイミング

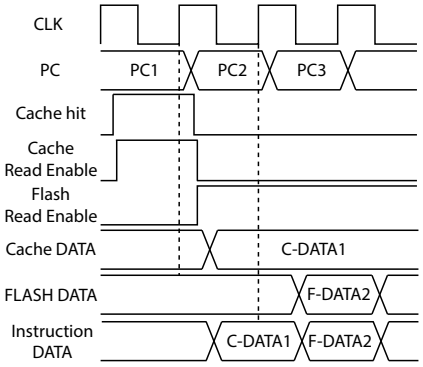


図 3 キャッシュ導入後の命令メモリアクセスタイミング

る必要がある．本研究では命令キャッシュのラインサイズを 4 ワードに拡張し、その効果を評価することにした．

ここで、ラインサイズを 4 ワード化する場合、フラッシュメモリから 1 サイクルに 1 ワードしか読み込めない事から図 4 のように 4 つのバッファを設け、4 サイクル間に各ワードを蓄えておき、キャッシュのラインに書き込む必要がある．したがって、2.1 節で説明したフラッシュメモリにアクセスしないサイクルにフラッシュメモリのアドレスを繰り上げ、次の命令列を 1 ワード読み込み、バッファに格納する．このよう動作でバッファに 4 つのワードが溜まったら命令キャッシュのラインにその 4 ワードを書き込み、次のラインの命令列をバッファに蓄えていく．

また、バッファに格納されたデータの中からヒットする場合（Buffer hit）も存在する．その場合は、ロジック側がバッファから命令列を読み出せるようにした．

しかし、4 つのワードがバッファに揃う前に、新たにキャッシュミスが発生すると、そのミスに対するラインの命令列をバッファに書き込まなければならない．したがって、その前までバッファに蓄えていた命令列は、4 ワードの命令列が揃う前に命令キャッシュに書き込まなければならない．

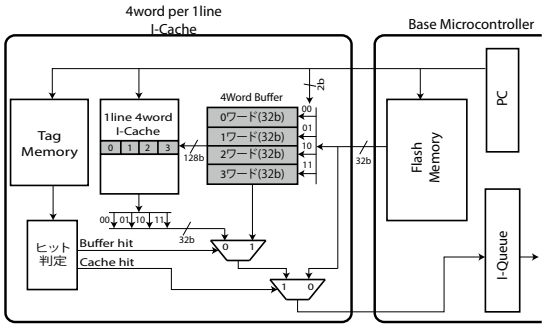


図 4 ラインサイズを 4 ワード化した場合の命令キャッシュ周りのブロック図

本研究ではこの問題を解決するために 1 つのラインの各ワードに対する有効ビット（4bit）をつけ、4 ワードの命令列が揃う前にキャッシュに書き込む時は、バッファに揃った命令列ワードにだけ有効ビットをセットし、その他は無効にするようにした．したがって、あるラインの無効ワードにアクセスしようとする場合、キャッシュミスになり命令キャッシュラインが新たに更新されることになる．しかしそのラインに有効なワードが存在していた場合、その有効ワードは破棄されてしまう可能性があるため、命令キャッシュの 1 ライン 4 ワード化後、ヒット率が下がる可能性もある．

3. 命令キャッシュの実装

本研究では、以上の構成に基づき、ベースマイコンおよび命令キャッシュ導入後のマイコンを RTL 実装した．RTL 実装を行った理由は、消費電力を評価するためであり、RTL 実装による詳細な消費電力評価方法は 4 章で説明する．本章では、RTL 実装を行った命令キャッシュの具体的な設計に関して説明する．

3.1 1 ライン 1 ワード命令キャッシュ

前章でキャッシュ導入後の命令メモリアクセスタイミングを説明したが、キャッシュミスの直後にヒットする場合は、図 5 のような問題が発生する．キャッシュヒット時には、ヒットしたサイクルでキャッシュ Read Enable 信号（図 5 の I-Cache Read Enable）を与え、キャッシュメモリからデータを読み出すが、キャッシュミス時には、ミスした次のサイクルでフラッシュメモリからデータを読み出せる為、キャッシュミスの次のサイクルでキャッシュ Write Enable 信号を与えなければいけない．したがって、図 5 のように、キャッシュミスの直後のサイクルでキャッシュヒットした場合、命令キャッシュへの Read アクセスと命令キャッシュへの Write アクセスが重複してしまう問題が発生する．

本研究ではこのような問題を解決するために、以下の 2

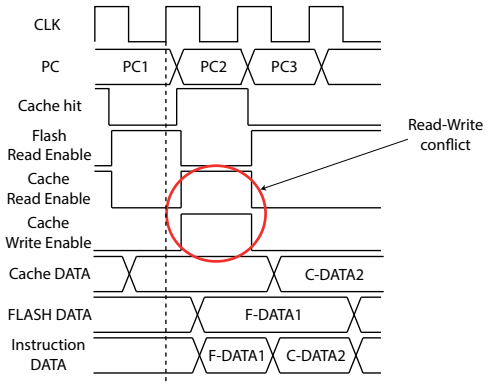


図 5 キャッシュミスの直後にキャッシュヒットする場合の動作タイミング

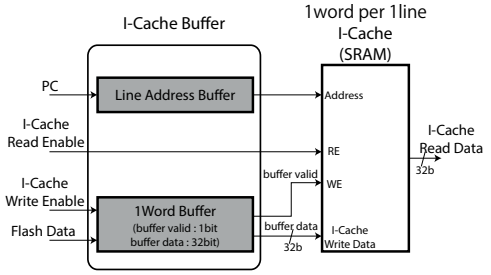


図 6 バッファ追加方式のブロック図

つの方式を用いることにした．

(1) **サイクル分割方式**
ミスの後ヒットする場合のみ、1 つのサイクルの前半で命令キャッシュへ Write アクセス、後半で命令キャッシュへ Read アクセスする方式

(2) **バッファ追加方式**
図 6 のように命令キャッシュの入力側にバッファを設け、ミスの起きた場合、キャッシュにアクセスせずバッファに命令列を蓄えておき、次のミス時にバッファのデータを書き換え、バッファに蓄えていたデータはキャッシュメモリに書き込む方式

サイクル分割方式は、バッファを用いないため、バッファ用のレジスタが不必要であり、レジスタの数を少なくすることができるため、ヒット判定回路の規模がより小さくなる．しかし、サイクルを 2 分割し、前半で命令キャッシュへの Write アクセス、後半で命令キャッシュへの Read アクセスを行わなければならないため、サイクル周期を既存のサイクルより長くする必要がある．したがって、動作周波数が遅くなる短所がある．バッファ追加方式は、サイクルを分割する必要はないが、バッファを用いるため、バッファ用のレジスタが必要になり、ヒット判定回路の規模がサイクル分割方式より大きくなることが短所である．

3.2 1 ライン 4 ワード命令キャッシュ

4 ワード化行う場合、サイクル分割方式では、1 サイクルで 4 ワードをフラッシュメモリから読み出し命令キャッシュに書き込めないため、バッファを用いるバッファ追加方式を拡張し、4 ワード化することにした．4 ワード化した命令キャッシュのブロック図を図 7 に示す．前章で説明した通り、フラッシュメモリにアクセスしないサイクルが存在する．このようなサイクルで、プログラムカウンター（PC）の値を各サイクルに 1 つずつ繰り上げ、図 7 の 4 つのバッファに、1 サイクル 1 ワードずつ蓄えていく．このような方式でバッファに、全ワード（4 ワード）が揃ったら、キャッシュ書き込み信号（cache write）を与え、その 4 ワードをキャッシュのラインに書き込む．

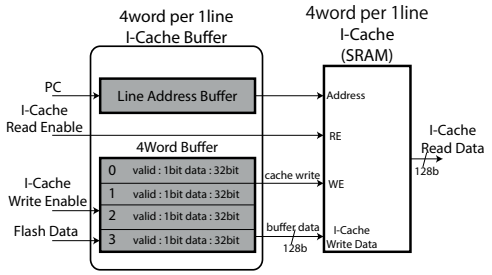


図 7 1 ライン 4 ワードキャッシュのブロック図詳細

4 ワードが揃う前に命令キャッシュに書き込む場合もキャッシュ書き込み信号（cache write）を与え、命令キャッシュのラインにバッファのデータを書き込む．この時、各バッファの有効ビット（計 4bit）は Tag メモリに書き込むようになる．

4. 消費電力評価

本章では、ベースマイコンに本研究で提案する命令キャッシュを導入し、消費電力の評価を行う．消費電力の見積もりは、RTL 記述からチップ設計データを生成し、バックアノテーション結果より消費電力を見積もった．

具体的には、メモリ以外のロジックで消費される電力は、ベースマイコンとキャッシュメモリ制御部を RTL 設計し、TSMC 0.18um CMOS セルライブラリを用いて実際のチップ設計データを生成し、消費電力見積もりに必要なデータ（ネットリスト、遅延情報、寄生容量情報、スイッチング率）を得て消費電力を見積もった．ここで論理合成、配置配線、消費電力見積もりに使用したツールは、表 1 の通りである．なお、ベースマイコンとしては 78K0R の公開アーキテクチャ情報を参考にしているものの、詳細アーキテクチャの違いや回路化手法・合成条件の違いなどにより、実際のマイコンの消費電力値とは食い違っている可能性がある．

表 1 論理合成・配置配線・消費電力見積もりに用いたツール

論理合成	Synopsys Design Compiler
配置配線	Cadence Encounter
消費電力の見積もり	Power Compiler

また、今回の評価において、マイコンの内蔵データメモ

リ、命令キャッシュメモリ、Tag メモリとして用いられる SRAM の消費電力は、以下の方法で見積もった．後術する評価プログラムのシミュレーションから、SRAM アクセス率（データメモリはデータメモリアクセス率、命令キャッシュメモリは命令キャッシュアクセス率、Tag メモリは、ヒット判定率（＝フラッシュメモリアクセス率＋キャッシュメモリアクセス率）を求め、0.18um プロセスにおける SRAM 消費電力の参考資料 [9] を基に、SRAM アクセスに消費される電力を見積もった．

一般的にフラッシュメモリ搭載マイコンの全体の消費電力（メモリ以外のロジック、データメモリアクセス、フラッシュメモリアクセスで消費される電力）の中で、フラッシュメモリアクセスに消費される電力が大きな割合を占めていることが知られている [2]．実際には、フラッシュメモリのメモリ容量や内部回路構成により、フラッシュメモリアクセスに掛かる消費電力が大きく異なってくることから、本稿では、フラッシュメモリアクセスに掛かる消費電力が全体の消費電力の 30 %、50 %、70 %を占めると仮定して、それぞれの場合における低電力化効果を評価する．

消費電力評価は、通常動作時（電源電圧：1.8V）での消費電力を見積もり、評価を行った．また、低消費電力化のために低電圧化を行った場合（低電圧動作時、電源電圧：0.9V）の消費電力評価も行った．但し、フラッシュメモリの場合は低電圧化を行うことが困難であるため、フラッシュメモリは低電圧化を行わず、1.8V の電源電圧で動作させるものとした．したがって、低電圧動作時には、フラッシュメモリの消費電力が占める割合はさらに大きなものとなる．また、通常動作時での動作周波数は、前章で説明した命令キャッシュの各方式で最も遅い動作周波数で動作する 1 ライン 1 ワードサイクル分割方式の最大動作周波数（後術）に合わせ、20MHz とし評価した．低電圧動作時の動作周波数は、電源電圧が 0.9V の場合における単体ゲートの回路シミュレーション結果から動作周波数を推定した．通常動作時と低電圧動作時における評価条件を表 2 にまとめる．

表 2 通常動作時、低電圧動作時においての評価条件

条件	通常動作時	低電圧動作時
CPU Core の電源電圧	1.8V	0.9V
SRAM の電源電圧	1.8V	0.9V
動作周波数	20MHz	4MHz

4.1 評価プログラム

評価プログラムとしては、以下の 6 つのプログラムを用いた．

- **Bubble sort**：12 個の数字で構成された任意の 5 個の数列に対し、Bubble 整列を行うプログラム（プログラムサイズ：614Byte）
- **C2F calculate**： 任意の 5 個の Celcius データを Fahrenheit に変換するプログラム（プログラムサイズ：585Byte）
- **Checksum**：Checksum を 5 回行うプログラム（プログラムサイズ：535Byte）
- **Copy verify**：Copy verify を 5 回行うプログラム（プログラムサイズ：602Byte）
- **Factorial**：任意の 5 個の数字に対する Factorial を行うプログラム（プログラムサイズ：606Byte）
- **EEMBC Coremark**：一般的にマイコンプロセッサでベンチマークプログラムとして用いられるプログラムであり、List processing（find, sort, remove/insert）、CRC アルゴリズムを 666 個のデータに対し、10000 回実行するプログラム（プログラムサイズ：11701Byte）

本研究で RTL 実装したベースマイコンは、78K0R のアーキテクチャを参考にしており、78K0R ソフトウェア開発環境（CubeSuite+）で生成されるプログラムバイナリコードを用いた RTL シミュレーションが可能である．したがって、上記評価プログラムのプログラムバイナリコードを生成し、RTL シミュレーションを行うことで消費電力の評価を行った．

4.2 消費電力・性能評価

本節では、論理合成・配置配線後のチップ設計データから最大動作周波数と消費電力を見積もった結果を示し評価する．本稿では、命令キャッシュのサイズが 128Byte〜4KByte での評価を行った．本節では、命令キャッシュサイズが 2KByte の場合における消費電力の評価を行い、次節で命令キャッシュサイズの変化による消費電力の評価を行う．

命令キャッシュ導入前と命令キャッシュ導入後の各方式におけるロジックのチップ設計データから、最大データ到達時間を求め、予想最大動作周波数を見積もった．その結果を表 3 に示す．

表 3 各方式における最大動作周波数見積もり（電源電圧：1.8V）

	最大動作周波数 [MHz]
キャッシュ導入前	約 45.45
サイクル分割方式	約 20.01
バッファ追加方式	約 41.67
1 ライン 4 ワード	約 40.00

サイクル分割方式では、ヒット判定による遅延に加え、1 サイクルの中で命令キャッシュへの Write アクセスと Read アクセスを両方行うため、動作できるサイクル周期も長くなる．したがって、キャッシュ導入前の最大動作周波数より 1/2 倍以下になる．バッファ追加方式では、ヒット判定による遅延の分、最大動作周波数が遅くなる．1 ライン 4 ワード方式では、1 ライン 1 ワードバッファ追加方

式に比べ、ヒット判定回路が複雑になるため、ヒット判定による遅延が長くなり、動作周波数がさらに遅くなる．

ここで示す結果は、Tag メモリを SRAM で実装する場合における結果である．Tag メモリは、レジスタとして実装することも可能であるが、前述した通り、本稿では、小型化に有利であり、低コストで実装できる SRAM を用いて評価を行った．しかし、Tag メモリとしてレジスタを用いる場合は、より高速にヒット判定ができるため、最大動作周波数が早くなると予想される．

命令キャッシュメモリのサイズが 2KB における各プログラムのヒット率を表 4 に示す．

表 4 各プログラムのヒット率（キャッシュメモリサイズ：2KB）

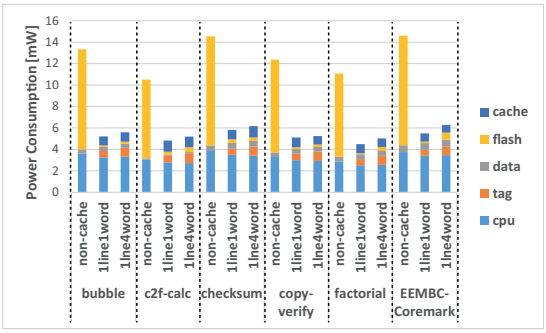
	1 ライン 1 ワード [%]	1 ライン 4 ワード [%]
Bubble sort	98.69	98.76
C2F calculate	96.76	96.11
Checksum	96.92	98.46
Copy verify	97.90	98.18
Factorial	98.63	95.67
Coremark	99.89	93.03

1 ライン 4 ワード命令キャッシュの場合、データの空間的局所性をさらに活かせるため、一般的にヒット率が上がる．しかし、前章で説明したように命令キャッシュの有効データが破棄されてしまう場合があるため、1 ライン 4 ワード化後、ヒット率が下がる場合も存在する．

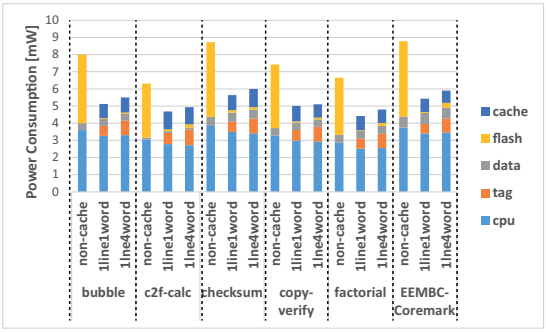
また、通常動作時において、フラッシュメモリ消費電力が占める割合による消費電力見積もり結果を図 8 に示す．ここで、1 ライン 1 ワード命令キャッシュの場合、サイクル分割方式とバッファ追加方式におけるヒット率が同一であり、消費電力見積もり結果もほぼ同一であったため、バッファ追加方式における消費電力見積もり結果だけ示す．

以上の結果から、ヒット率が高い程、フラッシュメモリの消費電力が大きく減ることが分かる．また、1 ライン 4 ワードの場合、Tag メモリおよびヒット判定回路が 1 ライン 1 ワードのキャッシュメモリより複雑になるため、Tag 部の消費電力が 1 ライン 1 ワードより大きい．そして、フラッシュメモリの消費電力割合が大きい程、命令キャッシュによるフラッシュメモリの消費電力削減量も大きくなるため、低電力化効果が大きくなる事が分かる．フラッシュメモリの消費電力が占める割合が以下の表 5 に示す割合以上であると命令キャッシュによる低電力化効果が期待できる．

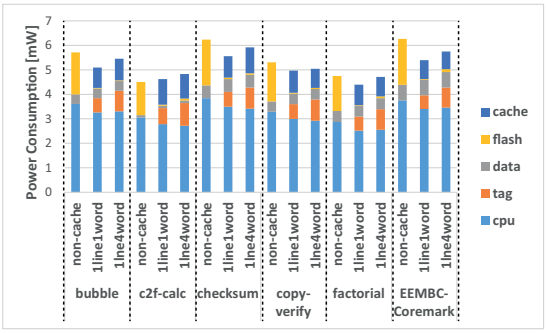
ここで、低消費電力化のためにフラッシュメモリ以外の電源電圧を 0.9V に低電圧化した場合、以下の消費電力式 1[1]（C_L：負荷容量、V_{dd}：電源電圧、f_{clk}：動作周波数、貫通電流、リーク電流は無視できる程小さいものとする）から、ロジック、データメモリ（SRAM）、Tag メモリ（SRAM）で消費される電力は、電源電圧が 1/2 倍、動作



(a) フラッシュメモリ消費電力割合が 70 %の場合



(b) フラッシュメモリ消費電力割合が 50 %の場合



(c) フラッシュメモリ消費電力割合が 30 %の場合

図 8 通常動作時の各プログラムの電力評価結果

表 5 低電力化効果が得られるフラッシュメモリ消費電力割合クロスオーバーポイント

	1 ライン 1 ワード [%]	1 ライン 4 ワード [%]
Bubble sort	21.6	26.5
C2F calculate	31.8	35.1
Checksum	21.2	24.5
Copy verify	25.0	26.2
Factorial	24.3	29.4
Coremark	18.6	23.3

周波数が 1/5 倍になったことで、通常動作時の消費電力の約 1/20 倍になる．一方、フラッシュメモリの消費電力は、電源電圧は変わらず、動作周波数が 1/5 倍になるため、通常動作時の消費電力約 1/5 倍になる．

$$P \propto C_L {V_{dd}}^2 f_{clk} \quad (1)$$

したがって、低電圧動作時での、フラッシュメモリの消費電力が占める割合は、通常動作時にフラッシュメモリの消費電力の割合が 70 %であった場合は約 90 %、50 %で

あった場合は約 80 %、30 %であった場合は約 63 %に増える．

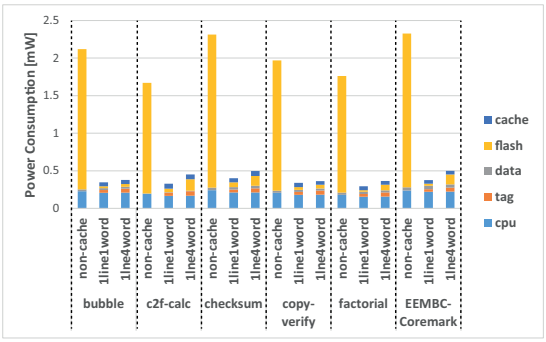
また、図 9 に、フラッシュメモリ消費電力割合による電力評価結果（低電圧動作時）を示す．これは、命令キャッシュ（SRAM）を低電圧化しているため、命令キャッシュの低電力化効果がさらに上がることやフラッシュメモリの消費電力の割合が大きくなったことが大きな原因であり、フラッシュメモリ搭載マイコンを低電圧で動作させた場合、命令キャッシュ導入による低電力化効果がさらに上がることが分かる．

以上の結果をまとめ、図 10 に、EEMBC-Coremark プログラムにおける通常動作時と低電圧動作時での低電力化効果（命令キャッシュ導入前の消費電力と命令キャッシュ導入後の消費電力の比）を示す．フラッシュメモリの消費電力が占める割合が大きい程、低電力化効果も大きくなり、低電圧動作時でさらに低電力化効果が大きくなる事が分かる．通常動作時と比べ、低電圧動作時における低電力化効果は 1.5〜2 倍に上がった．

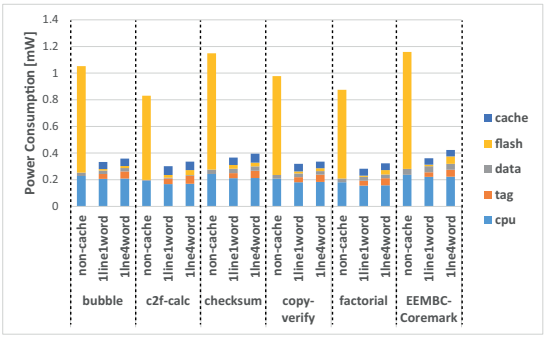
4.3 命令キャッシュサイズ変化による消費電力評価

命令キャッシュサイズが十分に大きいと 1 ライン 1 ワードの場合でも高いヒット率が得られるため、1 ライン 4 ワード化を行ってもヒット率や消費電力は改善はされない（1 ライン 4 ワード命令キャッシュは、Tag メモリ部の回路が複雑になるため、ヒット率がほぼ同一の場合、1 ライン 1 ワードの命令キャッシュより消費電力が大きくなる結果となった）．しかし、一般的に使われるキャッシュメモリのサイズは、小さい場合が多く、キャッシュメモリのサイズが小さくなるとヒット率が低下する可能性がある．図 11 に、命令キャッシュメモリサイズの変化による各方式におけるヒット率および消費電力の変化を示す．ここで、評価に用いたプログラムは、EEMBC Coremark プログラムであり、通常動作時で、フラッシュメモリアクセスに消費される電力の割合が 50 %での消費電力評価を行った．

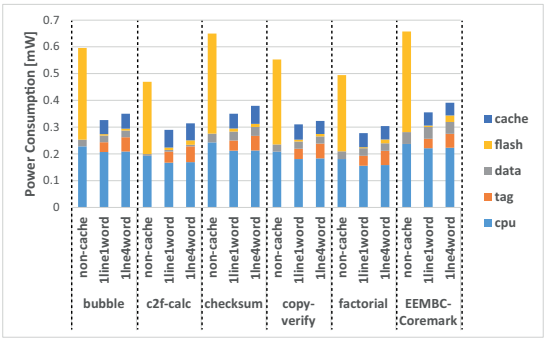
図 11 のように、命令キャッシュメモリのサイズが大きい場合（1〜4KByte）は、1 ライン 1 ワード命令キャッシュでもヒット率が高く、消費電力も 1 ライン 1 ワード命令キャッシュの方が小さい事が分かる．しかし、命令キャッシュメモリのサイズが一定のサイズ（512Byte）以下になると、1 ライン 1 ワード命令キャッシュのヒット率が大きく低下する．一方、1 ライン 4 ワード命令キャッシュのヒット率は比較的に少し落ちることが分かる．ヒット率が下がっていくと、図 11 の flash で示すように命令キャッシュによるフラッシュメモリ消費電力の削減はほとんど行われないことが分かる．以上の結果から、命令キャッシュメモリのサイズが大きい場合は、1 ライン 1 ワード命令キャッシュでも Tag メモリやヒット判定回路が簡単な 1 ライン 1 ワード命令キャッシュの消費電力の方が小さいが、キャッシュメモリサイズが小さい場合は、1 ライン 1 ワード命令キャッシュのヒット率が急激に低下するため、1 ライン 4 ワードの方が低電力化効果が大きいと言える．



(a) フラッシュメモリ消費電力割合が 70 %（通常動作時）の場合



(b) フラッシュメモリ消費電力割合が 50 %（通常動作時）の場合



(c) フラッシュメモリ消費電力割合が 30 %（通常動作時）の場合

図 9 低電圧動作時の各プログラムの電力評価結果

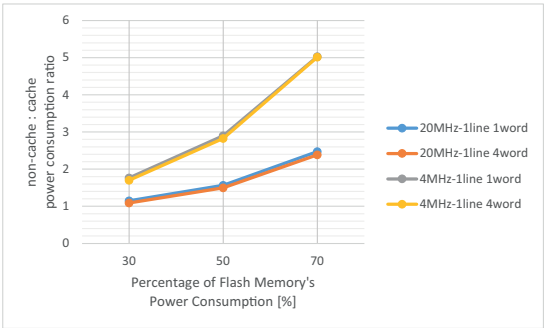


図 10 フラッシュメモリの消費電力が占める割合による低電力化効果、評価プログラム：EEMBC-Coremark

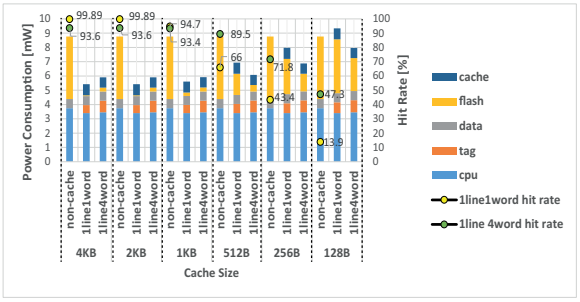


図 11 命令キャッシュサイズの変化によるヒット率および消費電力の変化グラフ（通常動作時、フラッシュメモリの消費電力が占める割合：50 %、評価プログラム：EEMBC Coremark）

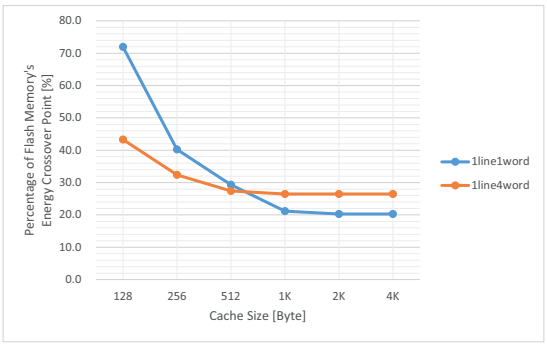


図 12 命令キャッシュサイズの変化による低エネルギー化クロスオーバーポイントの変化（通常動作時、評価プログラム：EEMBC Coremark）

これまで、命令キャッシュ導入によるフラッシュメモリ搭載マイコンの低電力化効果に関して述べてきた．ここでは、命令キャッシュ導入による最大動作周波数の変化（表 3 参照）を考慮した低エネルギー化効果に関して説明する（図 12）．1 ライン 1 ワード命令でもヒット率が高い 1〜4KByte では、低エネルギー化効果が得られ始める割合が 1 ライン 1 ワード命令キャッシュの方が低い（約 20〜21 %）が、命令キャッシュメモリサイズが 512Byte 以下の場合、その割合が 1 ライン 4 ワードより高くなり始め、キャッシュメモリが 128Byte では、72 %以上ではないと命令キャッシュ導入による低エネルギー化効果は得られないことが分かる．

5. まとめ

本稿では、フラッシュメモリ搭載マイコンの消費電力を減らすことを目的に、命令キャッシュを導入するアーキテクチャを説明し、その低電力化効果に関して述べた．

命令キャッシュ導入による消費電力評価の結果、マイコン全体の消費電力でフラッシュメモリの消費電力が占める割合が大きい程、命令キャッシュ導入による低電力化効果が大きいことが分かった．フラッシュメモリの消費電力が全体の 70 %を占める場合、命令キャッシュ導入前の消費電力より 1/2〜1/3 倍の消費電力が得られた．本稿で用いた評価プログラムにおいて、フラッシュメモリの消費電力が全体の 20〜30 %以上を占める場合は、命令キャッシュ導入による低電力化が期待できることが分かった．

ここで、フラッシュメモリ以外を低電力化した場合の低電圧動作時では、命令キャッシュ導入による低電力化効果がさらに上がり、通常動作時において、フラッシュメモリの消費電力が全体の 70 %を占める場合、命令キャッシュ導入前の消費電力より約 1/5〜1/7 倍の消費電力が得られた．最も低電力化効率が悪い場合のフラッシュメモリ消費電力が 30 %を占める場合は、命令キャッシュ導入前の消費電力より約 1/2 倍の消費電力が得られた．

また、命令キャッシュメモリのサイズが大きい場合は、1 ライン 1 ワード命令キャッシュにおいても高いヒット率が得られるため、1 ライン 4 ワード化による低電力化効果は得られなかったが、命令キャッシュメモリのサイズが小さくなると 1 ライン 1 ワード命令キャッシュのヒット率が大きく落ちるため、1 ライン 4 ワード命令キャッシュの低電力化効果が大きいことが分かった．

今後は、命令キャッシュのヒット率を上げるためのアーキテクチャ上のアイディアを加え、さらなる低電力化を目指すこと、より大規模で現実的なアプリケーションで評価すること、チップ実装を行いより精密な消費電力評価を行うこと、等を進める予定である．

参考文献

[1] A.P. Chandrakasan and R.W. Brodersen.: *Minimizing Power Consumption in Digital CMOS Circuits*, *Proceedings of the IEEE*, 83（4）:498 –523, Apr. 1995.
[2] 大場 浩司、河合 一慶、松下 留美、石原 国泰、江藤 公治.: 超低消費電力 *16* ビットオールフラッシュマイコン *78K0R/Kx3-L* の開発,
[3] 溝口 誠、石川 潔、中野 正隆、熊谷 敏幸、磯貝 英夫、田中 健太郎.: 新 *16* ビットマイコン *78K0R* シリーズ,
[4] ルネサスエレクトロニクス、*78K0R* マイクロコントローラ ユーザーズマニュアル　命令編,
http : *//documentation.renesas.com/doc/DocumentServer/r01us0029jj0600*$_7$*8k0r.pdf*
[5] ルネサスエレクトロニクス、*R8C/Tiny* シリーズソフトウェアマニュアル,
http : *//documentation.renesas.com/doc/products/mpumcu/rjj09b0002*$_r$*8csm.pdf*
[6] 総合開発環境 CubeSuite+,
http : *//japan.renesas.com/products/tools/ide/cubesuite*$_p$*lus/#.Up0jgCmwfIU*
[7] EEMBC Coremark,
http : *//www.eembc.org/coremark/download.php*
[8] RL78/Kx3-L アプリケーションノート/サンプルコード,
http : *//japan.renesas.com/products/mpumcu/78k/78k0rkx/78k0rkx3l/app*$_n$*otes.jsp#.Up8apb9uhjE*
[9] 3.1 On-Chip SRAM
http : *//www.csd.uoc.gr/ hy534/03a/s31*$_r$*am*$_b$*l.htm*