# 代表的な数値計算を用いた組込み向けCPUの演算性能の評価

川村　　暁・佐々木慶文・佐藤千太郎・丸岡　　章

# Performance Evaluation of Fundamental Numerical Operations on Embedded CPUs

**Satoshi KAWAMURA, Yoshifumi SASAKI, Sentaro SATO and Akira MARUOKA**

*Dept. of Information Technology and Electronics, Faculty of Science and Engineering, Ishinomaki Senshu University MIYAGI. 986-8580*

## Abstract

Actual performance of embedded CPUs could be affected by the system architecture, operating system and application development tools such as compiler. Executing some benchmark programs with prototyping systems is more effective way to obtain the actual performance of an embedded CPU rather than going through datasheets of the CPU. Therefore, we evaluated operating time of some typical numerical operations by executing benchmark programs on five CPU boards with different ARM core-based embedded CPUs, and clarified their performance characteristics.

## 1. はじめに

身の回りの様々な機器の高機能化は、それらの機器に組み込まれたコンピュータシステムにより実現されている。例えば、インターネットにつながる情報家電やパソコンと同等の機能を有するスマートフォンは好例である。これらの機器を開発する場合、全てをスクラッチで行うことは時間・コスト等の面からも適さないため、何らかのOS（Operating System）が搭載されることが多い[1]。最近では、汎用的なLinux等のOSを組込み向けの設定にして落とし込む手法が多々見られる。汎用的なOSの場合は、ある程度一般的に用いられているパソコン用のソフトウェアを動作させることも可能になる[(1)~(3)]（ネットワークにつながる情報家電やインターネットに接続できる携帯電話を見れば明らかである）。

このようなシステムに搭載される組込みCPUは、本来、実応用機器の制御を目的として開発されてきたものであることから、汎用的な計算機で用いられるCPUと異なる特性を有する場合が多い[(4)~(20)]。消費電力が少なく発熱が少ないこと（ファンなどの特別な冷却機構が無くとも動作するのが望ましい）、コストが低いこと（製品単価の上昇が許容されないため）、必要最低限の機能が搭載されていること（必要十分であればよい）、などである。選択の幅を広げるため、SoC（シリコンオンチップ）のような周辺装置を組み込んだCPUが用意されることも少なくない。また、情報家電やスマートフォンに見られるようなネットワーク機能まで持つ機器の場合、OSを搭載することが殆どであるため、これが容易な32 bit CPUが用いられることが多い。

組込み向けCPUには様々な種類がある（例：ルネサス系（SH系、H8系）・ARM系（ARM-7、9、11）等）が、32 bitの組込みCPUで見た場合、ARM系のCPUコアの市場占有率が70%を占めるとも言われる[(3)]。そこで本稿では、世代の異なるARM系CPUコアを搭載した組込みCPUに着目し、性能評価を試みた。筆者らは、これまでにも組込み向けCPUの性能評価を試みている[(17)~(20)]が、本稿ではFFTに代表されるような実用レベルの処理においては多数の数値演算が主となる点に着目し、特に、整数及び浮動小数点数の数値演算性能を測定した。さらに、ARM系CPUの演算性能について詳しく検討するため、コンパイラに与えるオプションや種々の高速化回路を有効にするコンパイラオプションについても実験を行った。これは、高速化に資する付加的機能・機構が利用できる場合はできるだけ利用することを意味する。実験結果の評価・比較のため、汎用PCに搭載されているIntel x86系CPUと同系列の組込み向けCPUであるATOM CPU

（ATOM 240：第一世代の ATOM）でも同様の実験を行い、その結果を評価基準（規格化した値：1.0 とする）として用いた。また、実験環境としては、実応用を考慮して、OS には Linux を、コンパイラには現在標準的に用いられている gcc 4 系列を用いた。

## 2. ARM 系の CPU について

ARM 系の CPU コアは、ARM ltd. が開発している ARM アーキテクチャに基づいた CPU コア群である[6]~[16]。ARM ltd. はコアアーキテクチャや IP コアの設計を行い、それらのライセンスを取得した企業が、それらの IP コアに基づく CPU を製造している。

ARM アーキテクチャは、モステクノロジー社の 8 bit CPU 6502（任天堂のファミリーコンピュータに搭載されていた CPU として有名）を 32 bit CPU に拡張するように（6502 に示唆を得て）設計された[6]、[7]。即ち、6502 とプログラムの互換性が取りやすいように考えられた CPU アーキテクチャである。これに加えて、当初から 32 bit RISC CPU として設計されていたこと、複雑な機構（トランジスタを多く必要とするキャッシュや複雑な CPU の高速化手法）がなく低消費電力であること、比較的高速であることなどの特徴があった（ARM2 はクロック周波数が同じ Intel 80286 より高速だったという指摘もある[7]）。

ARM 系 CPU は、命令セットアーキテクチャに基づく世代（ARM1～ARM11）により大別される[6]~[16]。ただし、同一世代に属する CPU でも、高速化を見据えた拡張命令（SIMD[2] 演算機構、ベクトル演算支援機構、浮動小数点演算支援機構、命令密度向上と実行効率を上げるための 16 bit 実行機構など）の有無、キャッシュの有無は異なっている（様々な特徴を持つ CPU が同一世代に混在している）。このため、支援機構の搭載／非搭載があること、製造元によって CPU の IP コアに手を加えられている可能性がある（ARM 系 CPU は、あくまで命令セットレベルでの互換性が維持できていればよい）ことから、性能の評価は容易ではない（同一クロックであっても性能が異なる）。このことは応用製品を開発する場合の CPU の選択を柔軟にしている（最適解が見つかる可能性）が、一方で、選択の困難性（どれが最適か判断しにくい）を高めているという側面を持つ。

拡張命令は有効に使いこなせれば演算処理速度を大幅に向上できる可能性があるが、現状では、コンパイラやライブラリの対応が追いついていない[3]。しかし、ARM 系 CPU においても世代が進むにつれて高速化を見据えた機構が標準で搭載されるようになってきている[8]~[16] ため、これら支援機構を用いた演算処理の実行速度を計測する必要がある。よって、gcc に備わる一般的な最適化オプションだけではなく、機器固有の機能を有効にするオプションの有無による影響についても検討した。実験の詳細は次章で記す。

## 3. 実験の方法

実応用を指向して演算性能を評価するため、以下の 3 つの場合に分けて実験を行った。いずれも、典型的な応用プログラムにおいて基本的に用いられる演算要素である。

実験 A：整数演算（加減乗除算および剰余演算）
実験 B：単精度浮動小数点演算（加減乗除算）
実験 C：単精度浮動小数点の三角・指数対数・超越関数の演算（sinf、 cosf、 atan2f、 atanf、 powf、 sqrtf、 logf、 三角関数は 0～$\pi$ の範囲で演算）

それぞれの演算を 10 万回から 100 万回まで 10 万回おきに演算回数を増やして実行し、その演算時間を測定した。その結果から、演算回数に対してはほぼ線形的に演算時間が増加することが明らかとなったため、100 万回演算時の演算時間に基づき 1 命令あたりの大凡の単位演算時間を導出している（ただし、ループ処理のための余分なインストラクションが含まれていることになる：インクリメントと条件付きジャンプ命令）。時間の計測には、clock( ) よりも計測精度のよい gettimeofday( ) 関数を用いた。計測プログラムは C 言語で記述し、gcc を用いてコンパイルした。ここで、それぞれのオペランド（加減乗除算であればふたつ、三角関数・指数対数関数であれ

ば一つ）の生成にはメルセンヌツイスタ[21]、[22]を用いて、偏っていない数（乱数）を与えた。

前章までに示したとおり、ARM系CPUのような組込み向けCPUは、整数演算はCPUに実装されていても、浮動小数点演算や数学関数の演算を担うハードウェア機能はCPUには実装されておらず、ライブラリによって実行される場合が多い。実験Bは、浮動小数点演算支援機能がハードウェアで実装されていない場合、同演算は全てコンパイル時にライブラリがマージされる（ライブラリプログラムにより計算される）。実験Cは、浮動小数点演算支援機能を搭載しているCPUでも浮動小数点の四則演算止まりである場合が多く、多くのCPUではライブラリに頼って演算することになる。さらに、各種演算の支援機構（浮動小数点演算、ベクトル演算など）が存在するため、通常のライブラリを用いた場合だけを計測しても、CPUの全能力を計測できていないことになる。そのため本稿では、gccで評価プログラムをコンパイルする際、以下の3通りで実験を行った。ただし、条件γについては各種演算の支援機構が搭載されておりかつgccで利用可能な場合に限られるため、いくつかのCPUについてだけ実験を行った。

条件α　gcc＋最適化オプションの場合：gcc std
条件β　gcc＋CPU固有オプション＋最適化オプションの場合：gcc tuned
条件γ　gcc＋CPU固有オプション＋各種演算支援機構＋最適化オプションの場合：gcc vfp or gcc neon

ここで、最適化オプション（-Ox）については、O0〜O3まで実験を行い、処理速度がもっと

表1　比較に用いたシステムの仕様。

(a) Armadillo-9

| System information of Armadillo-9 (Atmark Techno) | | |
|---|---|---|
| CPU | Chip | Cirrus Logic EP9315 |
| | Core | ARM920T(d,wb) rev 0 (v4l) |
| | Architecture | ARMv4T |
| | Freq. | 200MHz |
| | I cache | 16KB |
| | D cache | 16KB |
| | L2 cache | - |
| | Internal SRAM | - |
| | FPU | - |
| | BogoMIPS | 99.73 |
| System BUS | | 100MHz |
| Memory | | 64MB(60MB, 4MB Area is used as RAM disk.) SDRAM |
| Swap | | 0MB |
| OS | | 2.6.26-at4-sp93xx-port-pre2 |
| User Program ABI | | EABI |
| Compiler | | gcc version 4.3.2 (Debian 4.3.2-1.1) |

(b) Kurobox-PRO

| System information of Kurobox Pro (玄人志向) | | |
|---|---|---|
| CPU | Chip | Marvell 88F5182 |
| | Core | ARM926EJ-S rev 0 (v5l) |
| | Architecture | ARMv5TEJ |
| | Freq. | 400MHz |
| | I cache | 32KB |
| | D cache | 32KB |
| | L2 cache | - |
| | Internal SRAM | - |
| | FPU | - |
| | BogoMIPS | 266.24 |
| System BUS | | - |
| Swap | | 360MB |
| Memory | | 128MB DDR2 |
| OS | | Linux version 2.6.26-2-orion5x (Debian 2.6.26-24lenny1) |
| User Program ABI | | EABI |
| Compiler | | gcc version 4.3.2 (Debian 4.3.2-1.1) |

(c) Armadillo-440

| System information of Armadillo-440 (Atmark Techno) | | |
|---|---|---|
| CPU | Chip | Freescale i.MX257 |
| | Core | ARM926EJ-S rev 4 (v5l) |
| | Architecture | ARMv5TEJ |
| | Freq. | 400MHz |
| | I cache | 16KB |
| | D cache | 16KB |
| | L2 cache | - |
| | Internal SRAM | 128KB |
| | FPU | - |
| | BogoMIPS | 199.06 |
| System BUS | | 133MHz |
| Memory | | 128MB(102MB, 26MB area is used as RAM disk) LPDDR |
| Swap | | 0MB |
| OS | | Linux 2.6.26-at8 |
| User Program ABI | | EABI |
| Compiler | | arm-linux-gnueabi-gcc 4.3.2-1.1 (provided by Atmark techno) |

(d) Armadillo-500

| System information of Armadillo-500 (Atmark Techno) | | |
|---|---|---|
| CPU | Chip | Freescale i.MX31 |
| | Core | ARM1136JF-S |
| | Architecture | ARMv6TEJ |
| | Freq. | 400MHz |
| | I cache | 16KB |
| | D cache | 16KB |
| | L2 cache | 128KB |
| | Internal SRAM | 128KB |
| | FPU | VFP |
| | BogoMIPS | 399.76 |
| System BUS | | 133MHz |
| Memory | | 128MB DDR |
| Swap | | 0MB |
| OS | | Linux version 2.6.26-at6 |
| User Program ABI | | EABI |
| Compiler | | gcc version 4.3.2 (Debian 4.3.2-1.1) |

(e) BeagleBoard

| System information of BeagleBoard Rev. C4 (BeagleBoard.org) | | |
|---|---|---|
| CPU | Chip | OMAP3530DCBB72 |
| | Core | Cortex-A8 |
| | Architecture | ARMv7 rev 3 (v7l) |
| | Freq. | 720MHz |
| | I cache | 16KB |
| | D cache | 16KB |
| | L2 cache | 256KB |
| | Internal SRAM | 64KB |
| | FPU | VFP/NEON |
| | BogoMIPS | 484.21 |
| System BUS | | 165MHz (L3 Interconnection, u-boot console log) |
| Memory | | 256MB MDDR |
| Swap | | 957MB |
| OS | | Linux 2.6.29 |
| User Program ABI | | EABI |
| Compiler | | gcc 4.3.2-1.1 |

(f) ATOM (Aspire R3600-A34)

| System information of Aspire R3600-A34 (Acer.co.jp) | | |
|---|---|---|
| CPU | Chip | Intel(R) Atom(TM) CPU 230 @ 1.60GHz |
| | Core | Diamondville |
| | Architecture | Intel(R) Atom(TM) CPU 230 |
| | Freq. | 1600MHz |
| | Bus/Core Ratio | 12 |
| | I cache | 32KB |
| | D cache | 24KB |
| | L2 cache | 512KB |
| | FPU | MMX, SSE2, SSE3, SSSE3 |
| | Other | HT x86-64 |
| | BogoMIPS | 3200.27 |
| System BUS | | 533 MHz |
| Memory | | DDR2 800 MHz SDRAM/so-DIMM 1GB x 2 |
| Swap | | 4GB |
| OS | | Linux 2.6.31-22 SMP |
| Compiler | | gcc version 4.4.1 (Ubuntu 4.4.1-4ubuntu9) |

表 2　コンパイルオプション。ATOM については、原稿執筆時の gcc（4.5 未満）では ATOM 固有オプションが未対応だったため、std と tuned で同一オプションとした。

| Compile options | | |
|---|---|---|
| Common | | -O0, -O1, -O2 or -O3 |
| Armadillo-9 | std | |
| | tuned | -mtune=arm920t -march=armv4t -mfloat-abi=hard |
| Kurobox Pro | std | |
| | tuned | -mtune=arm926ej-s -march=armv5te -mfloat-abi=hard |
| Armadillo-440 | std | |
| | tuned | -mtune=arm926ej-s -march=armv5te -mfloat-abi=soft |
| Armadillo-500 | std | |
| | tuned | -mtune=arm1136jf-s -march=armv6j -mfloat-abi=soft |
| | vfp | -mtune=arm1136jf-s -march=armv6j -mfpu=vfp -mfloat-abi=softfp |
| BeagleBoard | std | |
| | tuned | -mtune=cortex-a8 -march=armv7-a -mfloat-abi=soft |
| | vfp | -mtune=cortex-a8 -march=armv7-a -mfpu=vfp -mfloat-abi=softfp |
| | neon | -mtune=cortex-a8 -march=armv7-a -mfpu=neon -ftree-vectorize -mfloat-abi=softfp |
| ATOM | std | |
| | tuned | |

も高速になった場合の計測値を採用した。

演算結果を比較するために、汎用 PC やサーバに用いられている x86 CPU のうち、組込み機器を指向した CPU である Intel ATOM を搭載した PC で計測した結果を規準とし、以下に示す性能指標を定義した。すなわち、式(1)に示す通り ATOM の性能を 1.0 とし、CPU のクロック周波数比と測定した演算速度の比（実行時間の比）の差を求めることにより、各組込み CPU の演算性能の特性（どのような演算に対して秀でているか）を検討した。

$$\text{性能の評価値} = \frac{ATOM[MHz]}{\text{対象の}CPU[MHz]} - \frac{\text{対象の}CPU\text{での実行速度[sec]}}{ATOM\text{での実行速度[sec]}} \quad \cdots\cdots\text{式(1)}$$

なお、 gcc は ATOM に対して固有最適化オプションが未対応であり（gcc 4.5 以降でサポートされる）、条件 $\beta$ での測定ができなかったため、今回は条件 $\alpha$ で計測した結果をすべての評価の基準としている。

## 4. 計算機実験の結果と考察

表 1 の CPU について、表 2 のコンパイル条件を用いて計算機実験を行った結果を示す。なお、すべての指標（規準）となる ATOM については、gcc での固有最適化オプションが未対応であ

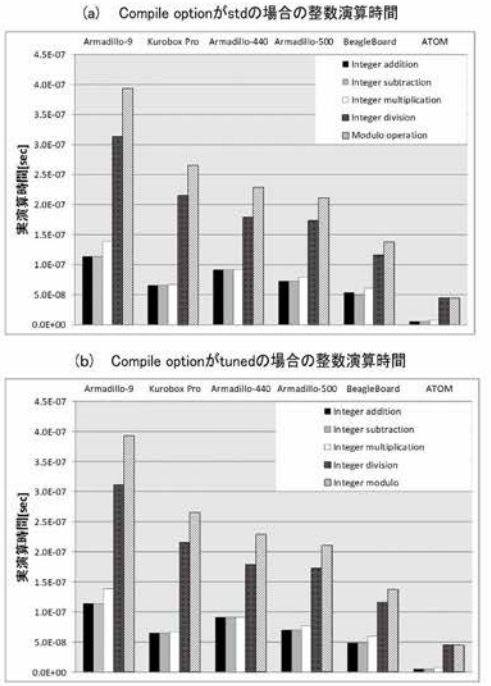


図 1　整数演算における、演算 1 回あたりにかかった時間。

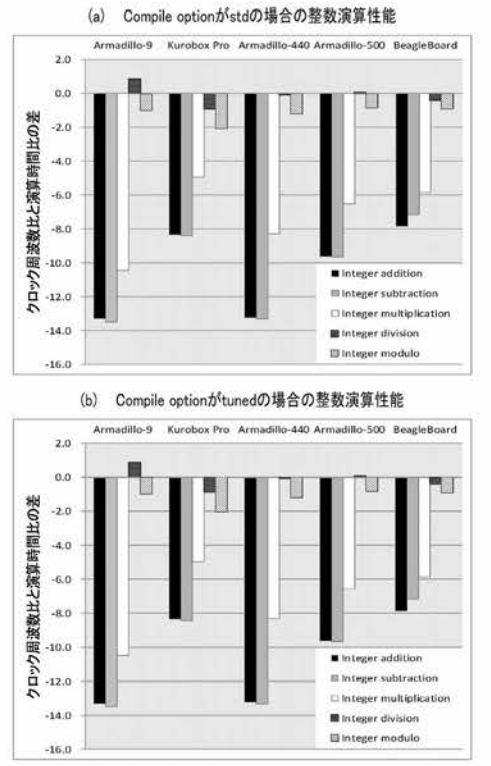


図 2　整数演算の性能比較。ATOM を規準とした式(1)の値。

るため、std と tuned のコンパイルオプションは同一としたため、計測結果も同値である。

### 4.1　整数演算：実験 A、 条件αおよび条件β

整数演算の結果を図 1 に示す。演算時間は 1 命令あたりの単位演算時間である。 CPU のクロック周波数の順位に従って時間がかかっている様子が分かる。また、基本的な演算の反復の場合、条件αから条件βへ変更した（より詳細な、最適なコードを生成するようなコンパイルオプションを与えた）ことによる影響は殆どみられない。

次に、式(1)により与えられる性能の評価値の結果を図 2 に示す。これも前図と同様、条件変更による影響はあまり見られない結果となった。また、他の実験（浮動小数点演算、関数演算）と比較して、評価値のマイナス幅が小さかった。ハードウェアの実装コストの低い整数演算の性能は、組込み向け CPU でもあるレベルで実装されていると考えられる。

### 4.2　浮動小数点演算：実験 B、 条件αおよび条件β

浮動小数点演算の結果を図 3 に示す。演算時間は 1 命令（または 1 要素の演算）の単位演算時間で示している。CPU のクロック周波数の順位に従って時間がかかっている様子が分かる。さらに、条件αから条件βへ変更した（より詳細な、最適なコードを生成するようなコンパイルオプションを与えた）事による影響が見られる結果となった。即ち、処理速度が若干改善されているということである。浮動小数点演算がライブラリ関数で計算されること、ライブラリもプログラムであるからコンパイラがコードを最適化したことが改善の要因として考えられる。

次に、式(1)により与えられる性能の評価値の結果を図 4 に示す。前図の結果がよりわかりやすく示されている。即ち、ATOM との差を示す指標値が縮小している（マイナス幅が縮小している：たとえば Armadillo-9 の浮動小数点減算は条件αだと約-47 であるのに対し、条件βだと

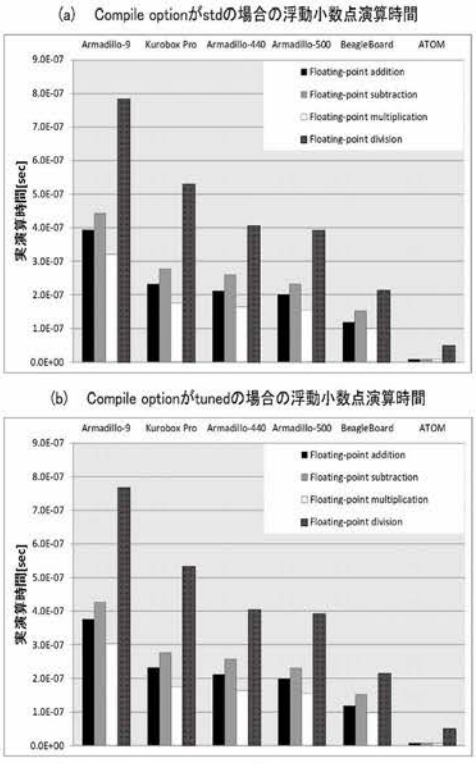


図 3　浮動小数点演算における、演算 1 回あたりにかかった時間。

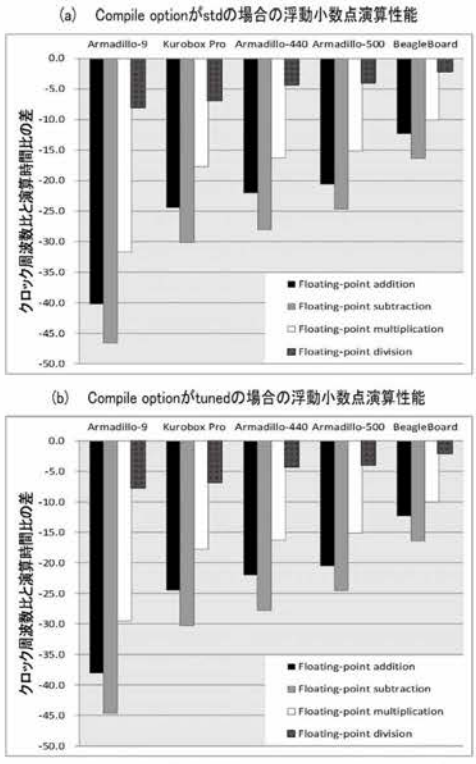


図 4　浮動小数点演算の性能比較。ATOM を規準とした式(1)の値。

−44.5 であり、2.5 程改善が見られる)。また、他の実験（整数演算、関数演算）と比較して、評価値のマイナス幅が大きかった。これは、浮動小数点演算を実現するライブラリ（≒プログラム）は規模が大きく実行にはコストがかかること、ATOM にはある程度の浮動小数点演算機構が搭載されているため比較的高速に実行できることから生じたと考えられる。

### 4.3 ライブラリ関数（三角・指数対数・超越関数）の演算：実験 C、 条件 $\alpha$ および条件 $\beta$

三角・指数対数・超越関数演算の結果を図 5 に、式(1)により与えられる性能の評価値の結果を図 6 に示す。演算時間は 1 命令（または 1 要素の演算）の単位演算時間で示している。CPU のクロック周波数の順位に従って時間がかかっている様子が分かる。条件 $\alpha$ から条件 $\beta$ へ変更した（より詳細な、最適なコードを生成するようなコンパイルオプションを与えた）事による影響は殆どみられない。これは、比較の基準点として用いている ATOM でもライブラリを使用している可能性が考えられる。また、浮動小数点演算ほどではないが、評価値のマイナス幅は比較的大きかった（図 6)。やはり、ライブラリを主体とする演算の場合、FPU に代表されるような演算を支援するハードウェアの存在の有無が大きく影響することがわかる。

### 4.4 演算支援機構（vfp、 neon）の影響：実験 A〜C、条件 $\gamma$

実験に用いた Armadillo-500 や BeagleBoard に搭載された ARM 系 CPU には、数値演算を高速化するためのハードウェア支援機構が搭載されている。そこで、支援機構を有効にした場合の影響について考察した。

整数演算、浮動小数点演算および三角・指数対数・超越関数の結果をそれぞれ図 7〜図 9 に示す。

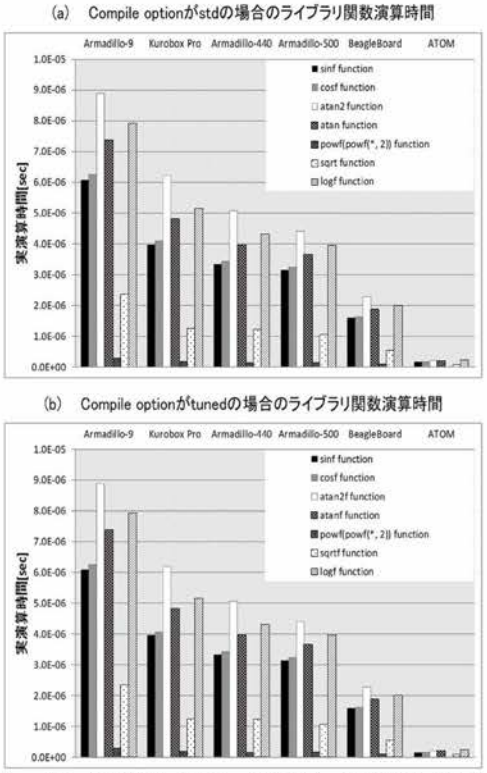


図 5 ライブラリ関数（三角・指数対数・超越関数）演算における、演算 1 回あたりにかかった時間。

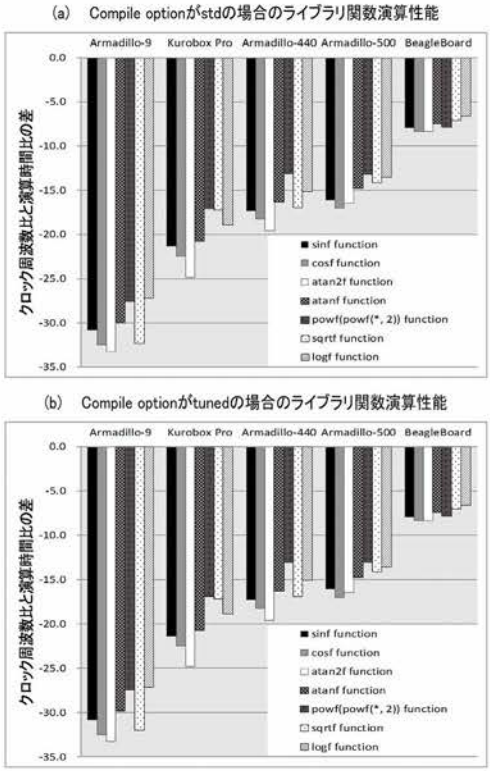


図 6 ライブラリ関数（三角・指数対数・超越関数）演算性能比較。ATOM を規準とした式(1)の値。

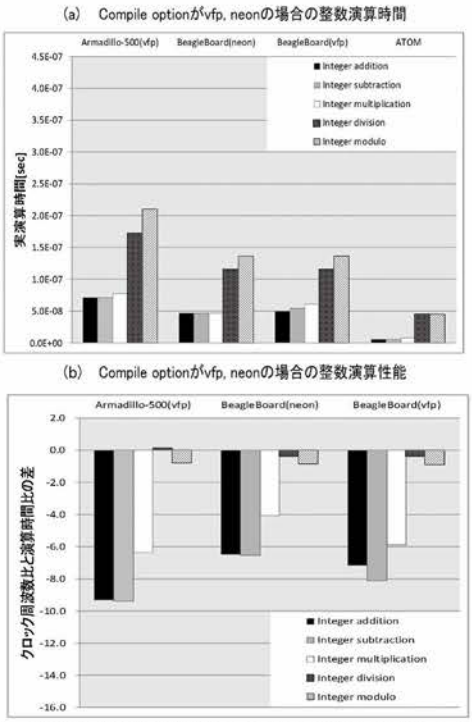


図7　数値演算を支援し高速化する機構を用いた場合の、整数演算時間と性能比較。

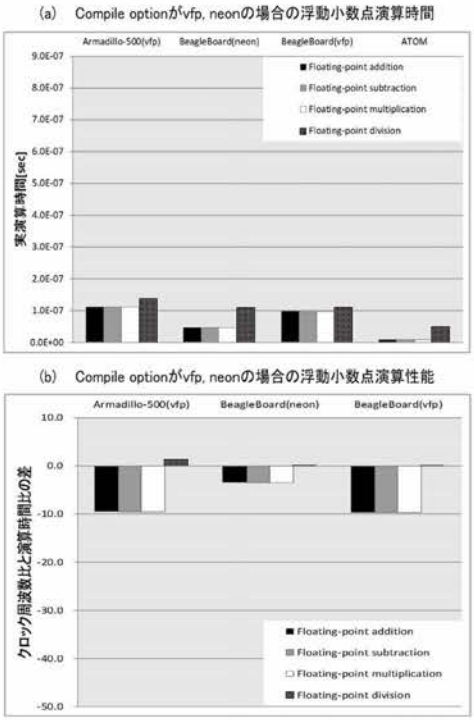


図8　数値演算を支援し高速化する機構を用いた場合の、浮動小数点演算時間と性能比較。

整数演算の結果は、標準の場合（図1および図2）と比較して、特にneonを有効にした場合に高速化される（評価指標のマイナス幅が縮小している）ことがわかる。これに対してvfpでは、若干の高速化はなされる（評価指標のマイナス幅が若干縮小）が、neonほどの劇的な変化は見られない。

浮動小数点演算の結果は、通常の場合（図3および図4）と比較して、非常に高速になっていることが分かる。評価指標のマイナス幅が数分の一となっていることからも分かるように、同一の処理を行った場合でも、処理速度が数倍に向上していることが分かる。この効果はvfpよりもneonを用いた場合に著しい。

最後に、ライブラリにより実現されていると推測される各種関数の結果について検討した。通常の結果（図5および図6）と比してある程度高速化されているが、浮動小数点演算ほどの効果は見られない。これは、ライブラリとは関数の演算自体がプログラムとして記述されたもの（オブジェ

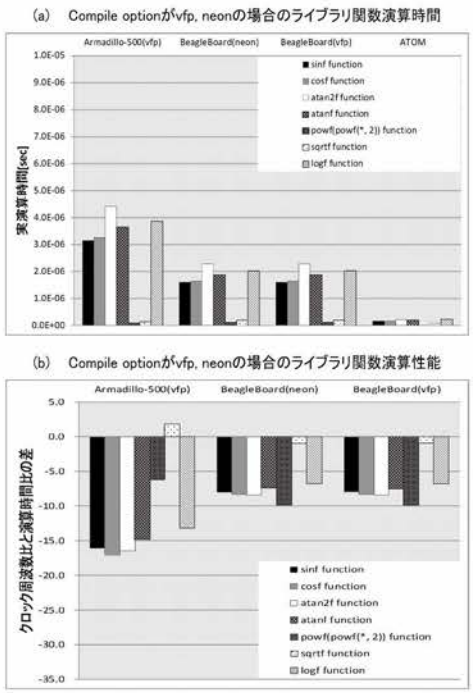


図9　数値演算を支援し高速化する機構を用いた場合の、ライブラリ関数（三角・指数対数・超越関数）演算時間と性能比較。

クト）であることから、単なる演算程の効果がなかったことが伺われる。逆に考えれば、それぞれの高速化支援機構により適応したライブラリを作成（記述）することにより、高速化できる可能性があることを示唆する結果ともいえる。しかしながら、特定の CPU アーキテクチャに依存するようなコーディングは、ソフトウェアパイプラインなどを考慮した上でアセンブラ言語により記述を行わなければならず、記述作業には大きな困難を伴うことは特記する。

## 5. まとめ

身の回りの組込み機器でよく利用されている ARM 系 CPU について、実応用を指向した各種演算の処理速度について詳細な検討を加えた。CPU の機能を利用するだけの処理と比べて複雑な処理をしなければならなくなるタスク（浮動小数点演算や各種の数学関数演算）は、ライブラリを用いた演算になる関係からどうしても処理が遅くなること、演算速度は CPU のクロック周波数から見積もることが出来ないこと、世代による影響も顕著であることなどが分かった。同時に、CPU に搭載されている数値演算高速化機構を有効に利用できた場合、浮動小数点演算・関数演算ともに大きく処理速度を向上できることも示された。また、コンパイラの最適化オプションをチューニングすることでも、ライブラリ関数の実行においても、若干ではあるが処理速度を向上できることも示した。実応用の場合、様々な厳しい条件があることを考えると意義のある結果と考える。また、実製品を構築する上で、同一世代の CPU でも速度が大きく異なる事を示した点で本稿の知見は意義があると思われる。

また、今回の実験では、単体の演算について評価を行ったが、積和演算などの複合的な演算や高速フーリエ変換のようなアプリケーションモジュールとして使われる処理などについても評価を行い、実アプリケーションを構築するためのさらなる知見を蓄積していくことが重要であると考えている。

## 文献

(1) 株式会社富士通ラーニングメディア著（2008）標準テキスト 組込みプログラミング《ソフトウェア基礎》、技術評論社
(2) 高田 広章 著、戸川 望編さん（2008）、組込みシステム概論（組込みシステム基礎技術全集 vol. 1）、CQ 出版
(3) 岡村佳郎（2007）Appendix x86 CPU とメモリ、OS を適切に使おう、p.70-p.73、Interface、Aug 2007
(4) David A. Patterson, John L. Hennessy 著, 成田 光彰訳（2006）、コンピュータの構成と設計～ハードウエアとソフトウエアのインタフェース 第 3 版（上）、日経 BP 社
(5) David A. Patterson、John L. Hennessy 著、成田 光彰訳（2006）コンピュータの構成と設計～ハードウエアとソフトウエアのインタフェース 第 3 版（下）、日経 BP 社
(6) 中森 章（2010）マイクロプロセッサ変遷史／1990 年代～2000 年代、p.44～p.63、Interface、2010 年 10 月号、CQ 出版
(7) 二宮 均（1996）最新 RISC CPU 事情（5）低消費電力 RISC ARM シリーズの概要、p.145-p.152、Interface、1996 年 6 月号
(8) Arm ltd.(2004) ARM アーキテクチャリファレンスマニュアル, ARM ltd.
(9) Arm ltd.（2008）ARM Architecture Reference Manual -- Arm v7-A and Arm-v7-R edition, ARM ltd.
(10) Arm ltd.（2007）ARMv7-M アーキテクチャアプリケーションレベルリファレンスマニュアル, ARM ltd.
(11) Arm ltd.（2002）ARM926EJ-S(Rev.0)テクニカルファレンスマニュアル, ARM ltd.
(12) Arm ltd.（2001）ARM7TDMI(Rev.4)テクニカルファレンスマニュアル, ARM ltd.
(13) Arm ltd.（2000）ARM9TDMI(Rev.3)テクニカルファレンスマニュアル, ARM ltd.
(14) Arm ltd.（2000）ARM920T(Rev.1)テクニカルファレンスマニュアル, ARM ltd.
(15) Arm ltd.（2000）ARM940T(Rev.2)テクニカルファレンスマニュアル, ARM ltd.
(16) Arm ltd.（2002）ARM1136JF-S / ARM1136J-S (リビジョン.r0p2) テクニカルファレンスマニュアル, ARM ltd.
(17) Satoshi Kawamura, Takahiro Nakanishi, Hitoaki Yoshida, Kazuaki Oozeki, Kazuo Fujimaki and Ryuji Gotoh (2006), Studies on the accuracy of numerical operations with embedded CPUs, IEICE

Electronics Express,　Vol. 3, No. 8, pp.149-155
(18) 川村　暁、廣井　富（2007）組み込み向け CPU の数値演算における超越関数演算の精度についての一考察、石巻専修大学研究紀要、第 18 号、pp.1〜6
(19) 川村　暁、廣井　富、村上　武、吉田等明、三浦　守（2008）組み込み向け CPU の三角関数演算における精度比較、石巻専修大学研究紀要、第 19 号、pp.9〜16.
(20) 川村　暁、佐々木慶文、廣井　富、村上　武、吉田等明、三浦　守（2009）組込み向け CPU の総合的な性能評価の試み、石巻専修大学研究紀要、第 20 号、pp.7-16
(21) Mersenne Twister Home Page, http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/mt.html
(22) M. Matsumoto and T. Nishimura (1998), Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator, ACM Trans. on Modeling and Computer Simulation Vol. 8, No. 1, January pp.3-30

## 註

1　ネットワーク機能をスクラッチで実装するのは、コーディングの量が膨大になること、安定動作するかどうかの試験が非常に難しい（作業量が多い）こと等からあまり行われない。既に定評のある Linux 等に備わるネットワーク機能を用いる場合が多い。

2　SIMD とは Single Instruction Multiple Data の略で、複数データについて一つの演算命令で同一の演算を行う演算方式のこと。データ密度の向上により、演算速度の向上が期待できる。適応しやすい例として、画像データや音声データの処理があげられる。

3　機械語（マシン語）を使えば使用可能であるが、コーディングが非常に難しいこと、CPU 毎に記述し直さなければならないこと、移植性・可読性が極端に悪くなるため推奨されない。