ALTERA ボードを使った VM の動作について

森脇淑也[†] 田中和明[††]

Ruby は、他のプログラミング言語と比較して、動的型付け（データ型の指定や宣言が不要）や豊富なライブラリなどの特徴があり、動作の本質的な部分に不必要なコードを書く必要がほとんどないためプログラムが理解やすく、効率的に書くことができるプログラミング言語である。しかし、メモリ使用量が多いため、既存の Ruby では組み込みシステム開発に用いることができない。そこで、開発中の組み込みシステム向け Ruby（軽量 Ruby）を用いることで開発効率を上げることを目的とする。現在は、軽量 Ruby を FPGA という PC 上で回路の書き換えが可能な LSI 上で動作させることで、開発効率の向上についての検証を行っている。

Toshiya Moriwaki[†] Kazuaki Tanaka[††]

Ruby is a easy to understand and efficient programing language than C language and Java language and so on, because of dynamic typing and rich method library. But Ruby uses a lot of memories to execute its program, so it can’t use as programing language for embedded systems. Now, new Ruby for embedded systems have been developed and my study is efficiency of the development using it. Concretely, New Ruby have been verify in FPGA

1. 序論

近年、半導体の急速な発達によるマイクロプロセッサの高性能化、低価格化にともない、組み込みシステムの適用範囲が急速に拡大している。組み込み市場の拡大により、企業間の競争が激化し、組み込み開発のコスト削減や開発期間の短縮といった開発効率の向上が求められるようになっている。また、ハードウェア開発に比べソフトウェア開発はコストが低く、柔軟性が高いため、システム全体に対する組み込みソフトウェアの開発比率が年々高まっており、ソフトウェア開発の大規模化、複雑化が起こっている。（図 1.1 を参照）

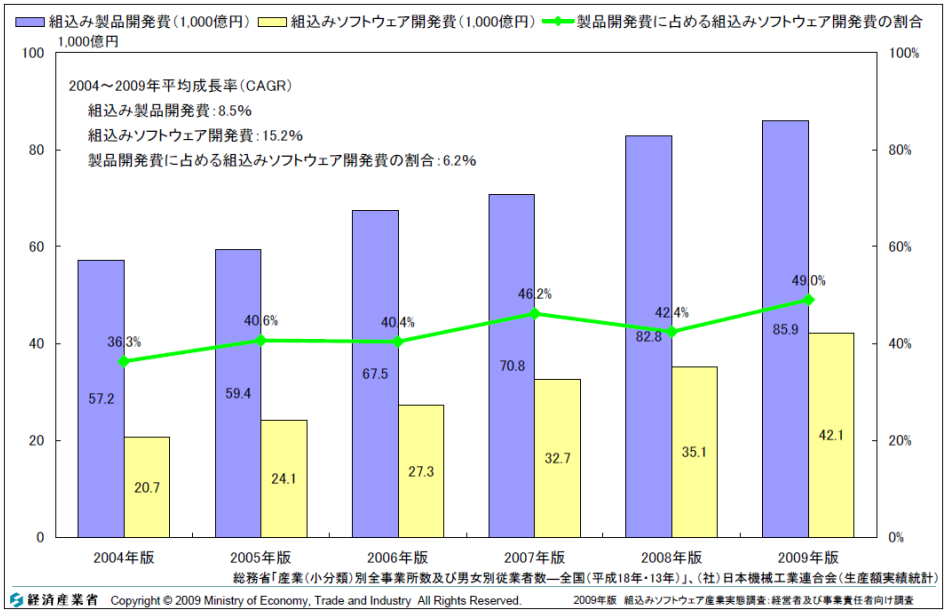

図 1.1 組み込み製品開発費と組み込みソフトウェア開発費の推移

† 九州工業大学大学院
Graduate School of Information Engineering, Kyushu Institute of Technology
†† 九州工業大学
Kyushu Institute of Technology

現在、組み込みシステムの開発に使用されているプログラミング言語 89%は、C 言語または C++言語である。（図 1.2 を参照） C 言語や C++言語などの高級言語は、アセンブラ言語に比べて人間が理解しやすく、開発環境に依存せず移植性がある。また、処理ごとに分割してプログラムする構造化プログラミングにより可読性や保守性が高い。

しかし、上にも書いたようにソフトウェア開発の大規模化、複雑化や、コスト削減や開発時間の短縮によって、より開発効率のよいプログラミングが必要となってきている。そこで、本研究の目的である生産性の高いといわれているプログラミング言語 Ruby を C 言語と併用して組み込み開発に用いることで、組み込みシステムの開発効率をより高めることができると考えた。

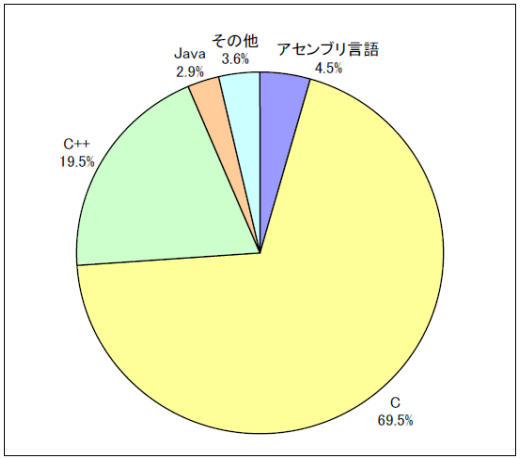

図 1.2 2010 年度組み込み開発で使用されているプログラミング言語（記述行数比率）

Ruby が生産性の高いプログラミング言語と言われている理由としては、ライブラリが充実しており、変数宣言の必要がなくプログラミングを行うことができるコンパクトさと、内容の同じ処理に対して何通りもの書き方でプログラムできる自由度の高さがあげられる。そのため、メンテナンス性や生産性を高め、より開発効率を良くすることができる。

また、C 言語と併用して組み込みシステムの開発を行うことで、以前作っていた資源も使用することができ、開発効率を高めることができる。しかし、現在の Ruby ではリソースの使用量が多く、メモリ容量の小さい組み込みデバイスでは、Ruby のプログラムを使用することができない。

そこで、現在開発中である従来の Ruby に比べ実行時のリソースが小さく組み込みデバイスでも使用できる軽量 Ruby を実験に用いた。軽量 Ruby システムは、現段階で開発途中であり、まだ組み込みシステムの開発では用いられていない。本実験では、現在できている軽量 Ruby を組み込みデバイス上で動作させる RiteVM を用いて、組み込みデバイス上で正常に動作を行うか検証を行った。

2. 研究目的・研究内容

2.1 研究目的

本研究では FPGA デバイス上で、組み込みシステム開発向けの Ruby（軽量 Ruby）で書かれたプログラムを実行することで、軽量 Ruby の動作についての検証を行うとともに、組み込みシステム開発で主流となっている C 言語、C++言語で書かれたプログラムとの比較を行い、軽量 Ruby の開発効率について検証を行うことを目的とする。

組み込みシステム開発における開発効率の高さを判断する基準は一概には言えないが、本研究では、コーディング時の開発期間に重点をおいて検証を行っていきたいと思う。現在の組み込み開発に求められることは高機能化、低コスト化、開発期間の短縮、品質の向上である。高機能化でコーディングの開発期間を短くできると全体の開発期間を短くでき、また開発期間は、システム開発を行う上で開発コストに直接関係し、品質の向上についてもコーディングの開発期間を短くすることで、設計、テストの時間の増大を行うことができ、品質の向上が見込める。現在組み込み開発に求められている多くの要因に開発期間は関係しており、開発期間は開発効率を上げるうえで非常に重要な要因だと考えられる。そのため、開発時間に関係しているコーディング時の開発期間の検証を行う。

2.2 研究内容

研究内容としては、軽量 Ruby のコンパイラがまだ開発されていなかったため、始めは開発されていた RiteVM が組み込みデバイス上で正確な動作を行うか確認するために Rite 中間表現プログラムを作成し、RiteVM の検証を行った。

本実験で行った検証内容は、

（検証 1）中間表現での C 言語の関数呼び出し

C 言語で関数を呼び出先の値と引数を足して呼び出し先にその値を返す関数と、処理時間を測定しはじめる関数と、処理時間を測定し終え計測した処理時間を表示する関数を作成し、実際に動作させた。

現在の組み込み開発では C 言語でのシステム作成が主になっている。そのため、C 言語のリソースが豊富にあり、C 言語のリソースを使うことが開発効率を高める上で必要不可欠になってくる。そこで、中間表現上で C 言語で作成した関数を呼び出すことで、RiteVM が C 言語のリソースを使用することができるか検証を行った。

（検証 2）中間表現上での繰り返しのプログラム
　オペコードを用いて繰り返しのプログラムを作成し、そのプログラムが正確に動作を行うか検証を行った。繰り返し回数は、1000 回から 10000 回まで 1000 回ずつ繰り返し回数を増やしていき、検証を行った。
　繰り返しのプログラムは、時間によって変化する値を制御する時や、外部からの入力に対して適切な動作を行う時など幅広い分野で使用される非常に基本的で大切なプログラムである。その為、目標とする値まで繰り返しを行うプログラムを作成して検証を行った。

（検証 3）中間表現の処理時間、処理速度
　処理時間や処理速度は、性能を評価する指標の 1 つとして重要である。組み込みシステムの場合は、処理時間や処理速度によって機能に大きな差が出る。繰り返しのプログラムを実行して処理時間、処理速度を計測することで RiteVM の性能について検証を行った。処理時間を測定する関数を用い、プログラムの処理時間を（検証２）のプログラムでそれぞれ計測を行った。また、処理時間をオペコードの命令数で割ることによって処理速度を求めた。

　現在は軽量 Ruby 専用のコンパイラが開発されたため、軽量 Ruby 全体を通した FPGA 上での検証を行っている。現段階では、開発環境（Windows）上で軽量 Ruby ファイルをコンパイルし、FPGA デバイスにコンパイルして変換したファイルと実行環境を構成して軽量 Ruby ファイルの動作を行っている。

3.　システム構成

3.1　Ruby

Ruby とは、まつもとゆきひろ氏が開発したマルチプラットフォームな、オブジェクト指向スクリプト言語である。
Ruby の特徴としては、以下のことがあげられる。
・インタプリタ言語のため、コンパイルの必要がない。
・変数に型がなく、変数宣言が必要ないので、シンプルに文法を書くことが可能。
・ガーベジコレクションにより、メモリの管理が不要。
・オブジェクト指向なので、クラス、継承、メソッド、Mix-in などの機能、概念が存在。
・正規表現や例外処理の機能が存在。
・C プログラムやライブラリを呼び出す拡張モジュールを組み込むことが可能。
・フリーソフトウェアである。

3.2　軽量 Ruby

　軽量 Ruby は、既存の Ruby の長所を残し、組み込みシステム開発向けに軽量化、最適化を行ったプログラム言語である。既存の Ruby との違いは以下のものがあげられる。
・ライブラリ内の機能の縮小（必要に応じてミニマルライブラリ、スタンダードライブラリ、フルライブラリを使い分ける）
・コンパイル型言語（軽量 Ruby 専用のコンパイラを用いる）
・軽量 Ruby 専用の VM（RiteVM）上で実行を行う。
　また、軽量 Ruby の動作手順を、.mrb 拡張子の時を図 3.1 に示す。

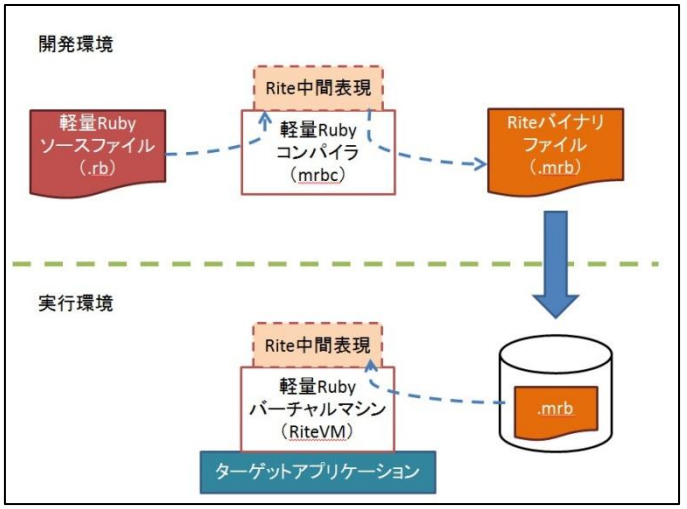

図 3.1 軽量 Ruby の手順

1、 開発環境（Windows、Linux など）上で Ruby ソースファイルを専用のコンパイラを用いて、Rite バイナリファイルに変換する。
2、 実行環境上に Rite バイナリファイルとライブラリ、RiteVM など動作に必要なファイルを追加する。
3、 実行環境上で、RiteVM を用いて Rite 中間表現を実行する。

3.3 FPGA

FPGA とは、**F**ield **P**rogrammable **G**ate **A**rray の頭文字をとったもので、現場（Field）で、書き換え可能（programmable、プログラム可能な）、LSI（論理ゲート（Gate）が格子（Array）状に並んでいるセミカスタム LSI）のことであり、簡単に言うと後からでも回路の書き換えが可能なロジックデバイスである。

FPGA の特徴としては、以下のような特徴がある。

・高速、小型、軽量、低消費電力。
・小さなロジック・モジュールを組み合わせて設計ができるため、設計の自由度が高い。
・PC 上で設計し、PC から FPGA に実装できるため開発期間が短い。
・設計の自由度の高さから、任意の論理回路の実現。
・多入力、多出力の並列処理が可能。
・プログラムを何度でも書き直すことができるため、柔軟に回路を変更することが可能。
・実装した回路の速度やゲートの消費効率などの回路の速度が設計者に依存。

本研究では、Altera 社が開発している Cyclone Ⅱ FPGA Starter Board を使用した。研究に使用したデバイスを図 3.2 に示す。

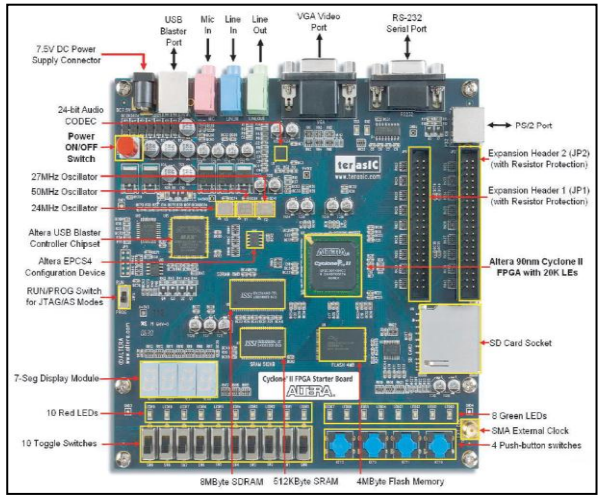

図 2.2　Cyclone Ⅱ FPGA Starter Board

Altera 社の Cyclone シリーズは、価格重視の量産アプリケーション向けに開発されている。ASIC（特定の用途のために設計、製造される集積回路）に匹敵するコストで高性能と低消費電力を両立させたデバイス。自動車、通信、家電、ビデオ処理、テスト、計測などの市場で幅広く利用されている。Cyclone Ⅱ は、Cyclone シリーズの第 2 世代である。他の 90nm 低コスト FPGA に比べて低消費電力、低コストが特徴である。

4. 結果、考察

4.1 結果（中間表現の検証）

検証のために作成したプログラム内容は、目標の繰り返し回数まで 1 から 1 ずつ足し算を行い、足した値を足されるたびに表示し、目標の回数だけ繰り返した後、処理開始から処理終了までの処理速度を求めて、表示するプログラムである。プログラムを実行させたときの実行結果を図 4.1 に示す。

```
998
999
1000
--Performance Counter Report--
Total Time: 3.15757 seconds  (157878474 clock-cycles)
+--------------+-----+----------+--------------+----------+
| Section      |  %  | Time (sec)|  Time (clocks)|Occurrences|
+--------------+-----+----------+--------------+----------+
method check
```

(1)　(2)　(3)

図4.1　実験プログラムの実行結果

図 4.1 の下線部(1)は、足した値が繰り返し表示されたことを示している。図 4.1 に表示されていないコンソー ルの上部には、“1” から “997” までの値が表示されている。図 4.1 の下線部(2)の Total　Time によって、処理時間“4.89134 seconds”が表示された。また、下線部(3)で“method check”が表示されている。

また、繰り返し回数を変えた時の処理時間と、処理速度のグラフを図 4.2、図 4.3 に示す。

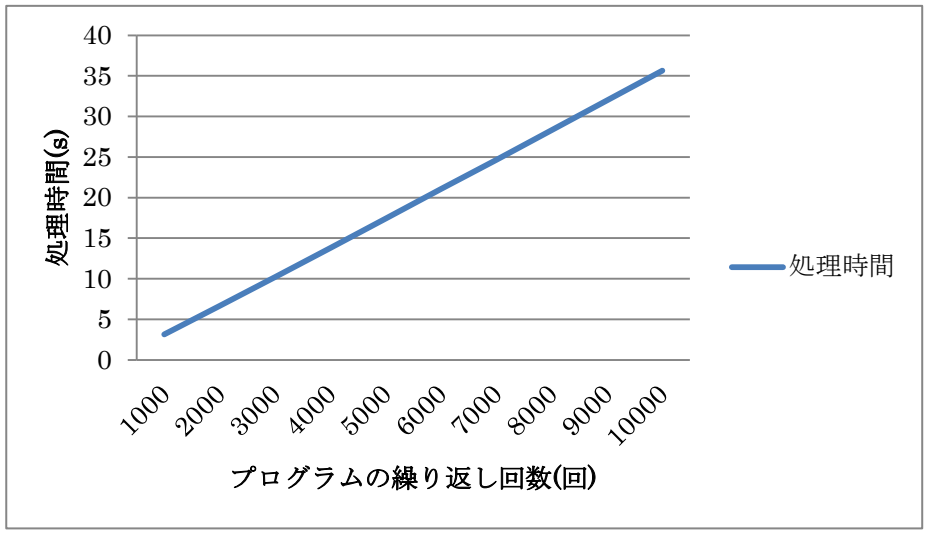

図 4.2　処理時間と繰り返し回数の関係

4.2　考察（中間表現の検証）

（検証1）の中間表現でのC言語の関数呼び出しは、正確に値が1ずつ足され表示されたことが図4.1の下線部(1)から分かり、また図4.1下線部(2)から処理時間も正確に表示され、図4.1の下線部(3)からmethod Checkが表示されていることがわかる。下線部(1)から(3)よりC言語で書かれた関数を、RiteVM上で正常に動作することがわかった。

（検証 2）の中間表現上での繰り返しのプログラムは、下線部(1)が正確に表示されたことから正しく動作を行ったと考えられる。しかし、Ruby は同じ内容のプログラムに対して何通りもの書き方ができるので、今回使わなかったオペコードを使用することにより違う方法での繰り返しを行うことができると考えられる。

（検証 3）の中間表現の処理時間、処理速度は、処理時間の方は図 4.2 から繰り返しの回数が増えても、それに比例して処理時間も増加していることがわかる。また、図 4.3 より命令数が増加するにつれ、処理速度は少しずつだが早くなっている。同じ回路上で C 言語を用い、同様の内容のプログラムを作成したところ 10000 回繰り返した時の処理時間は、20.729[s](図 4.4 に示す)であった。10000 回の時の中間表現プログラムの処理時間は、35.6558[s]であるので軽量 Ruby を用いたプログラムの処理時間は C 言語を用いたプログラムの約 1.72 倍である。これは、RiteVM で関数呼び出しを多くおこっているため、処理時間が余計にかかると考えられる。

4.3　結果（軽量 Ruby）

次に軽量 Ruby の試作版が完成したので、FPGA 上で軽量 Ruby が正しく動作を行うか検証を行っている。まず、Windows 上で軽量 Ruby を動作させたところ、gcc コンパイラのバージョンによっては正しく動作しないことが分かった。gcc コンパイラのバージョンが 3.4.4 のときに軽量 Ruby で make を行うとエラーが発生し、make を行うことができなかった。そこで、gcc コンパイラのバージョンを 4.6.1 に変更し、make を実行したところ正しく make を終了することができた。また、windows を make する際に make ファイルの一部を windows の実行ファイルである exe 拡張子に変更しないと正しく make を行うことができなかった。次に windows 上で Ruby ファイルを専用のコンパイラでコンパイルし.mrb ファイルに変換し、.mrb ファイルを実行したところ正しく動作を行わなかった。この原因は不明で現在調査中である。そこで、コンパイルをおこなって生成されるファイルを.mrb ファイルではなく、.c ファイルに変更し実行を行ったところ正しく動作を行った。

現在は FPGA 上に軽量 Ruby ファイル（.c ファイル）を動作させる環境を作成し、軽量 Ruby の動作の検証を行っている最中である。

4.4　考察（軽量 Ruby）

現段階では、軽量 Ruby はプログラムによっては正しく動作しないという事例があり、また軽量 Ruby は不完全で動作させることのできる環境をも限られている。そのため、まずは軽量 Ruby のプログラム上の問題を見つけ改善していくことが大切である。また、軽量 Ruby の詳細なリファレンスやサポートホームページがない段階なので、開発環境の作成にかなりの時間がかかり、開発言語の導入のしやすさはあまりよくないと言える。今後、リファレンスの作成や、サポートホームページの作成などを行い改善していくことで開発時間の短縮があげられると思われる。開発効率を高めるうえではこの部分の改善も大切であると考える。

5.　今後の展望

現在、FPGA 上で軽量 Ruby の動作を行うことができた。しかし、軽量 Ruby の動作を行う上で問題があるので、まずは軽量 Ruby の動作について既存の Ruby プログラムと同様の動作を行うのか検証を行っていきたい。軽量 Ruby の動作に問題がなくなってきたら、次に C,C++言語と軽量 Ruby の比較と FPGA デバイス上の機能に対する

検証をおこなっていきたい。C,C++言語と軽量 Ruby の比較内容としては、C、C++言語と軽量 Ruby 言語の違いによる開発期間の違いについて行っていき、FPGA デバイス上での検証については組み込みデバイスの各ハード（LED、7SEG など）の動作、メモリ使用量、実行時間の検証を行っていきたい。

参考文献

[1]高橋征義　後藤裕蔵,　たのしい Ruby　第 3 版
　ソフトバンク　クリエイティブ株式会社
[2]高田広章　枝廣正人　沢田　篤史　清水徹　中島達夫　平山雅之,
　組み込みシステム概論　CQ 出版社
[3]2010 年版組み込みソフトウェア産業実態調査
[4] 2009 年版組み込みソフトウェア産業実態調査