

XX-XX

卒業論文

LLVMコンパイラ基盤を用いた ベクトル化コード生成についての検討

2022年3月

永池 晃太郎

宇都宮大学工学部
情報工学科

内容梗概

日本語アブストラクト

Consideration of Vectorized Code Generation Using LLVM Compiler Infrastructure

Kotaro Nagaike

Abstract

ENGLISH abst

目次

内容梗概	i
Abstract	ii
目次	iii
1 はじめに	1
2 MIQS プロセッサ	3
2.1 RISC-V	3
2.2 MIQS 概要	3
2.3 ベクトル拡張付き RISC-V	3
3 LLVM	5
3.1 LLVM 概要	5
3.2 LLVM による自動ベクトル化機能	5
4 ベクトル拡張付き RISC-V コンパイラ	6
4.1 LLVM バックエンドにおける独自命令実装手法	6
4.2 ベクトル拡張付き RISC-V 命令の定義	6
5 評価と考察	7
5.1 生成されるアセンブリコードの評価	7
5.2 非対応命令についての考察	7
6 おわりに	8
謝辞	9
参考文献	10

第1章 はじめに

FPGA(Field Programmable Gate Array) はユーザによって回路の再構成が可能な LSI であり，目的の処理をハードウェアとして実装可能なデバイスである．近年 FPGA の大容量化，高性能化によって大規模な回路が実現可能になった．これにより FPGA は自動運転を始めとする組込み分野での利用増加が期待されている．

FPGA を用いたハードウェア開発は HDL(Hardware Description Language) による RTL(Register Transfer Level) 設計が広く用いられている．RTL は FPGA 上に構成する回路の信号の流れや制御構造を直接設計できる一方，動作検証やデバックが難しく，短期間で複雑な処理の開発は困難である．[1] そのため，FPGA による開発期間を短縮する方法として，専用ハードディスクとプロセッサを用いたソフトウェアによる処理を組み合わせる方法が考えられる．FPGA 上のハードウェアリソースを用いて実装するプロセッサのことをソフトコアプロセッサという．一般的に組込みシステムではそのコストやサイズ，消費電力などに制限があるためハードウェア資源に制約があることが多く，メモリバンド幅に限られることからメモリシステムの性能が高くないことが多い．メモリシステムの性能が低いと演算性能を高くしてもメモリアクセスに時間がかかり，結果としてシステム全体の性能はメモリシステムの性能によって左右される．[2] そこで，ソフトコアプロセッサによる処理を考える．すべての処理を専用ハードウェア回路で行うのではなく，高い性能の求められない汎用的な処理についてはソフトコアプロセッサにて行うことにより開発する専用ハードウェア回路を削減でき，ソフトコアプロセッサの性能が向上して専用ハードウェア回路を削減しても求められた性能に達することができればハードウェアの開発コストを抑えることができる．

組込み分野では AI 技術に注目が集まっている．独立行政法人情報処理推進機構の調査によると，将来強化/新たに獲得したい技術として組込み/IoT 関連企業の 46%が AI 技術を挙げている．[3]

AI 技術の応用としては画像認識などの画像処理が行われる．画像処理では画像を構成する画素に対して同じ処理を行うようなものが多い．このように複数のデータに対して同じ演算を行う処理については，単一命令で複数データの処理を行う SIMD(Single Instruction, Multiple Data) や，複数命令を並列に実行する MIMD(Multiple Instruction, Multiple Data) による並列処理で高速化が可能である．[?, ?] MIMD は複数の制御を並列化することができるため，異なる処理を同時に実行するアプリケーションでは有効である．しかし，プロセッサに複数の制御ユニットをもたせる必要があるた

め SIMD と比較するとハードウェアのコストが大きくなる．一方 SIMD は単一の制御ユニットで複数の演算ユニットを並列動作させるため制御ユニットのコストは低くなる．データ並列処理では複数のデータに対して同じ処理を実行するため SIMD によって並列処理が可能である．

SIMD による並列処理を行う場合，SIMD 命令は1命令で演算するデータ数が決まっている．そのため演算性能を上げるために同時演算数を変更すると機械語コードを作り直す必要がある．異なる同時演算数でも同一の機械語コードを利用可能とするためには，機械語コードが同時演算数に依存しないスケーラブルなベクトル拡張が必要である．スケーラブルなベクトル拡張により機械語コードを変更することなく，必要に応じて容易に同時演算数を増やし高性能化することが可能となる．

スケーラブルなベクトル拡張を実現したものとしてオープンな命令セットアーキテクチャである RISC-V をベクトル拡張したベクトル拡張付き RISC-V が提案されている．ベクトル拡張付き RISC-V は組み込み機器に広く用いられている ARM のベクトル拡張である ARM SVE(Scalable Vector Extension) の命令セットを参考に組み込み向けに RISC-V に拡張したものである．しかし，ベクトル拡張付き RISC-V に対応したコンパイラが存在していない．そこで解決策としてベクトル拡張付き RISC-V のベクトル命令のアセンブリコードを得るためのコンパイラの開発を検討した．

本論文では，第2章で現在の MIQS プロセッサについて述べる．第3章でベクトル拡張付き RISC-V 命令の生成のために利用したコンパイラ基盤である LLVM について述べる．第4章で実際に命令生成のための実装について述べる．第5章では実際のソースコードからアセンブリコードの出力を行った結果について述べる．

第2章 MIQS プロセッサ

2 章. 始まったばかり，頑張れ．

2.1 RISC-V

ここは 2.1 節です

図の挿入方法サンプル lena 画像を図 2.1 に示す．

2.2 MIQS 概要

ここは 2.3 節です．

2.3 ベクトル拡張付き RISC-V

ここは 2.3 節です．



図 2.1 lena さん

第3章 LLVM

3 章！ 黙々と書こう．

3.1 LLVM 概要

ここは 3.1 節です．

3.2 LLVM による自動ベクトル化機能

ここは 3.2 節です．

第4章 ベクトル拡張付き RISC-V コンパイラ

ここは4章です．終わるまで振り返るな ... 文の推敲は後でもできる．

4.1 LLVM バックエンドにおける独自命令実装手法

ここは4.1節です．

4.2 ベクトル拡張付き RISC-V 命令の定義

ここは4.2節です．

第5章 評価と考察

5章．終盤，もう少し．

5.1 生成されるアセンブリコードの評価

ここは5.1節です．

5.2 非対応命令についての考察

ここは5.2節です．

第6章 おわりに

論文のまとめと課題．お疲れ様でした．

謝 辞

本研究の機会を与えていただき，また，日頃から貴重な御意見，御指導いただいた，馬場 敬信教授，大津 金光准教授，大川 猛助教，横田 隆史教授，に深く感謝致します．そして，本研究において多大な御力添えを頂いた，Boaz Jessie Jackin 氏をはじめとする研究室，オプティクス教育研究センターの方々に感謝致します．

参考文献

- [1] 三好健文: “FPGA 向けの高位合成言語と処理系の研究動向,” コンピュータソフトウェア. Vol.30, No.1, pp.76-84, 2013.
- [2] Samuel Williams, et al.: “Roofline: an insightful visual performance model for multicore architectures,” Communications of the ACM. Vol.52, No.4, pp.65-76, 2009.
- [3] 独立行政法人情報処理推進機構 社会基盤センター: “2020 年度組込み/IoT 産業の動向把握等に関する調査,” 2021.
- [4] デイビッド・A・パターソン, ジョン・L・ヘネシー著 成田 光彰 訳: “コンピュータの構成と設計 第 5 版,” 日経 BP 社. 2017.