

SCHOOL OF COMPUTATION,
INFORMATION AND TECHNOLOGY —
INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis, Master's Thesis, ... in Informatics

Thesis title

Author

SCHOOL OF COMPUTATION,
INFORMATION AND TECHNOLOGY —
INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis, Master's Thesis, ... in Informatics

Thesis title

Titel der Abschlussarbeit

Author:	Author
Supervisor:	Supervisor
Advisor:	Advisor
Submission Date:	Submission date

I confirm that this bachelor's thesis, master's thesis, ... is my own work and I have documented all sources and material used.

Munich, Submission date

Author

Acknowledgments

Abstract

This study dives into the world of RISC-V custom extension development and its application to Tiny Machine Learning (TinyML). An important part in the development of RISC-V custom extension is to identify performance bottleneck on benchmark. With a clear view of the bottleneck, custom extension can be further refined. However, it remains to be a challenge since most of the existing open-source profiling tools don't support RISC-V architecture explicitly.

In this work a tool that supports interactive multi-level profiling on RISC-V architecture is developed. It integrates well into the ETISS toolchain, a RISC-V development virtual platform developed at the Technical University of Munich (TUM). Given an instruction trace generated by ETISS plugin feature, the tool collects information on-demand and outputs the callgrind format, which can be used in the GUI Kcachegrind for interactive usage.

The tool is put into practice for benchmarking muRiscvNN, providing use cases in real scenarios. It illustrates how the process of identifying performance bottleneck can be simplified and sped up. Besides, it provides an infrastructure for developers to extend the profiling functionalities based on their need.

Contents

Acknowledgments	iii
Abstract	iv
1 Introduction	1
1.1 Motivation	1
1.2 Task Definition	1
2 State of the Art	2
2.1 Virtual Platform	2
2.2 Profiling Tools	2
3 Background	3
3.1 RISC-V ISA	3
3.2 ETISS	3
3.3 ELF File Format	3
3.4 Machine Learning Deployment	3
3.5 Valgrind	3
4 Implementation	4
4.1 Approach	4
4.1.1 ELF File Information Extraction	4
4.1.2 Basic Block Construction	4
4.1.3 Callgrind Output Format Converter	4
Abbreviations	5
List of Figures	6
List of Tables	7

1 Introduction

1.1 Motivation

TinyML has obtained more attentions in academy as well as in industries recently. There is huge potential to revolutionize the world when hundreds of billions of embedded devices are able to run machine learning applications offline. Nevertheless, given the resource-constrained nature of embedded devices, it still remains a challenge to codesign hardware and software.

RISC-V is promising in TinyML applications. For one thing, traditional x86 architecture is not power efficient enough compared to RISC-based architecture. For another, Arm-based processors require licensing fees since they are vendor-specific, whereas RISC-V ISA is open-source. The TUM EDA chair has developed the RISC-V instruction set simulator ETISS, which features the plug-in functionalities and supports rapid development of customized ISA based on user needs.

This thesis aims to provide a methodology of profiling TinyML applications in ETISS at various levels of abstraction interactively. The tool is further demonstrated by identifying performance bottlenecks in muRiscvNN kernel library.

1.2 Task Definition

2 State of the Art

2.1 Virtual Platform

2.2 Profiling Tools

3 Background

3.1 RISC-V ISA

RISC-V, which stands for the fifth generation of reduced instruction set computer, was first developed by Krste Asanović in 2010 at UC Berkeley. Unlike ARM-based microprocessors, RISC-V is an open-source ISA. Hence, it has been considered "the Linux of microprocessors". As a matter of fact, RISC-V is a lot more than that. The groundbreaking innovation lies in its modular design.

For a long time, incremental ISA, including x86 and ARM, has been the industry convention. In other words, architects keep adding instructions but never removing for the sake of backward compatibility. This leads to tremendous complexity in chip design. Besides, hardware costs increase since more silicon areas are required for decode and execution of a wider variety of instructions. And most importantly, this monolithic design approach introduces less flexibility. There is the analogy - you pay for the whole buffet, but all you want is salad.

3.2 ETISS

3.3 ELF File Format

3.4 Machine Learning Deployment

3.5 Valgrind

4 Implementation

4.1 Approach

4.1.1 ELF File Information Extraction

4.1.2 Basic Block Construction

4.1.3 Callgrind Output Format Converter

Abbreviations

List of Figures

List of Tables