TransLib: A Library to Explore Transprecision Floating-Point Arithmetic on Multi-Core IoT End-Nodes

Seyed Ahmad Mirsalari*, Giuseppe Tagliavini*, Davide Rossi*, Luca Benini*†
*University of Bologna, Bologna, Italy, †ETH, Zurich, Switzerland

***Abstract*—Reduced-precision floating-point (FP) arithmetic is being widely adopted to reduce memory footprint and execution time on battery-powered Internet of Things (IoT) end-nodes. However, reduced precision computations must meet end-do-end precision constraints to be acceptable at the application level. This work introduces TransLib[1], an open-source kernel library based on transprecision computing principles, which provides knobs to exploit different FP data types (i.e., float, float16, and bfloat16), also considering the trade-off between homogeneous and mixed-precision solutions. We demonstrate the capabilities of the proposed library on PULP, a 32-bit microcontroller (MCU) coupled with a parallel, programmable accelerator. On average, TransLib kernels achieve an IPC of 0.94 and a speed-up of 1.64× using 16-bit vectorization. The parallel variants achieve a speed-up of 1.97×, 3.91×, and 7.59× on 2, 4, and 8 cores, respectively. The memory footprint reduction is between 25% and 50%. Finally, we show that mixed-precision variants increase the accuracy by 30× at the cost of 2.09× execution time and 1.35× memory footprint compared to float16 vectorized.**

***Index Terms*—transprecision computing, IoT end-nodes, parallel programming, SIMD vectorization**

I. Introduction

The support for less-than-32-bits floating-point (FP) formats (a.k.a. SmallFloats) is becoming an essential prerequisite in modern computing systems with strict requirements for accuracy and dynamic range [1]. The main benefits of adopting SmallFloats on Internet of Things (IoT) end-nodes come from two main aspects: (i) the reduction of the memory footprint for data and parameters: and (ii) the reduction of execution time by exploiting Single Instruction Multiple Data (SIMD) vectorization. These factors also positively impact the energy consumption of algorithms running on IoT end-nodes. A substantial advantage of reduced-precision FP data types is the ability to simultaneously perform a SIMD operation on multiple sub-word elements, resulting in a theoretical speed-up of 2× for 16-bit data compared to the 32-bit baseline. At the same time, vectorization of 16-bit data reduces the memory footprint.

Due to their flexibility, low power, and low cost, embedded microcontrollerss (MCUs) have become the most popular IoT computing platforms. However, the power budget and memory storage of MCUs are insufficient for deploying state-of-the-art machine learning (ML) models. In recent years, *transprecision computing* has emerged as an evolution of approximate computing. This paradigm allows fine-grained control over precision to enhance energy efficiency without compromising the overall quality of results [2] [3].

Since IoT applications require high performance and extreme energy efficiency in a power envelope of a few mW, near-threshold parallel computing platforms for IoT end-nodes have emerged. PULP [4] is a heterogeneous open-source system-on-chip (SoC) platform that includes an MCU and a programmable multi-core cluster. The PULP SoC equips an on-chip SRAM memory (L2), hosting resident code and application data, while the cluster cores share a low-latency Tightly Coupled Data Memory (TCDM). Both MCU and cluster cores support *Xpulpv2*, an ISA extension for energy-efficient near-sensor computing. On the software side, the PULP SDK [5] provides a complete software environment, which includes a compiler toolchain supporting SmallFloat types [6], a virtual platform [7], and the PULP-OS environment providing a low-level runtime library.

In this scenario, a transprecision library targeting a modern platform for IoT computing has several requirements, including an accuracy model for multiple FP types, support for vectorization, and support for parallel execution. We propose TransLib, an open-source library including kernels from two application domains widely represented on IoT end-node devices: signal processing and ML. The main contributions of our work are:

- The library includes Python golden models and C programs optimized for IoT end-nodes.
- The golden models support fixed and mixed-precision variants.
- Users can employ various error metrics, possibly adding custom ones.
- We provide a sequential version of the C code, various parallel versions (i.e., 2, 4, and 8 cores), and all their vectorized variants.

II. TransLib Design

In the first version, we included a set of kernels commonly employed in IoT for filtering, feature extraction, classification, and basic linear algebra such as CONV, MATMUL, FIR, DWT, KMEANS, SVM (LINEAR, and RBF), and FFT. Each kernel design includes a Python model and a C program. The Python model generates the input dataset, computes the kernel output as a golden reference, and assesses the accuracy using a customizable error metric. These golden models are built on top of PyTorch type system. TransLib supports homogeneous float,

[1]https://github.com/ahmad-mirsalari/TransLib

978-3-9819263-7-8/DATE23/© 2023 EDAA

TABLE I: Evaluating Transprecision. FP32 and FP16 stand for float and float16, respectively.

	MSE			Total Cycles			Memory Footprint			
	Fixed-precision		Transprecision	Fixed-precision		Transprecision	Fixed-precision		Transprecision	
Kernel	FP32	FP16	FP16,FP32,FP32	FP32	FP16	FP16,FP32,FP32	FP32	FP16	FP16,FP32,FP32	FP16,FP32,FP16
CONV	0	1.35E-04	5.55E-06	312008	183056	414406	37420	19932	28172	19980
MATMUL	0	5.36E-05	2.38E-06	504873	259217	570874	34212	18332	30212	24932
FIR	0	3.77E-04	8.65E-06	508454	296683	673254	10688	6596	8644	6796

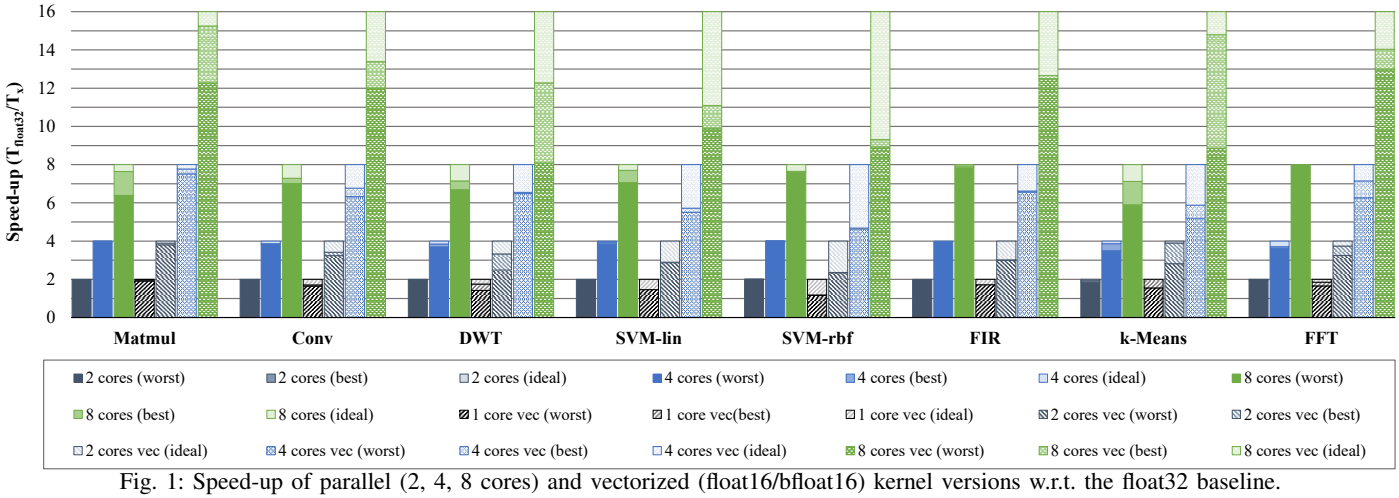

Fig. 1: Speed-up of parallel (2, 4, 8 cores) and vectorized (float16/bfloat16) kernel versions w.r.t. the float32 baseline.

float16, and bfloat16 types and also includes an extension to support mixed-precision computations. We introduced a set of flags to simulate different instructions typically available in the ISA extensions for IoT end-nodes (e.g., fused multiply-and-add rounding and custom casting behavior). The C program provides several code variants (sequential version with DSP optimizations, parallel version with low synchronization overhead, packed-SIMD vectorization of 16 bits FP types) to guarantee efficient execution on diverse end-node configurations. Overall, the code includes a set of optimizations portable among MCU-class targets and supporting both homogeneous and mixed-precision configurations. TransLib supports vectorization for half-precision FP types (e.g., float16 and bfloat16), and all kernels can execute in scalar or vectorial mode on a parametric number of cores. TransLib allows the user to check the output with multiple verbosity levels by employing several flags. Finally, a set of hardware per-core performance counters (e.g., total cycles/instructions, active cycles, shared memory contentions, FPU stalls, load stalls, branch stalls, and instruction cache misses) can be activated through a dedicated flag.

III. A Case Study: TransLib on PULP

Fig. 1 illustrates the speed-ups achieved executing on two, four, and eight cores, combining the benefits deriving from parallelism and vectorization. The suffix *vec* indicates the execution of the vector variant. The bars show the worst, best, and ideal speed-up values executed on all architectural configurations compared with a single-core scalar baseline. On average, we achieved 1.97×, 3.9×, and 7.6× speed-ups on 2, 4, and 8 cores with no vectorization. Furthermore, vectorization improves the speed-up – between 1.16× and 2×, while the accuracy in terms of mean squared error is roughly $1.35 * 10^{-4}$ (for float16), very close to the roundoff error of this format. Reduced-precision FP types also result in a 25% to 50% reduction in memory footprint.

TransLib supports mixed-precision variants for SVM, Conv, MATMUL, and FIR in the current version. Table I shows that transprecision variants lead to 2.09× longer execution times due to additional cast operations and 1.35× greater memory requirements than float16 vectorized, although they gain more accuracy by around 30×. As a key outcome of this exploration, designers can adopt this methodology to evaluate the best solution considering accuracy, throughput, and memory constraints for the target application. More information is available on TransLib's GitHub page.

Acknowledgement

This work was supported by the APROPOS project (g.a. no. 956090), founded by the European Union's Horizon 2020 research and innovation program.

References

[1] A. Sabbagh Molahosseini *et al.*, "Low-Precision Floating-Point Formats: From General-Purpose to Application-Specific," *Approximate Computing*, pp. 77–98, 2022.

[2] A. C. I. Malossi *et al.*, "The transprecision computing paradigm: Concept, design, and applications," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2018, pp. 1105–1110.

[3] G. Tagliavini *et al.*, "A transprecision floating-point platform for ultra-low power computing," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2018, pp. 1051–1056.

[4] M. Gautschi *et al.*, "Near-threshold RISC-V core with DSP extensions for scalable IoT endpoint devices," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, pp. 2700–2713, 2017.

[5] PULP Software Development Kit. Website: https://github.com/pulp-platform/pulp-sdk. Accessed: 2022-09-05.

[6] F. Montagna *et al.*, "A Low-Power Transprecision Floating-Point Cluster for Efficient Near-Sensor Data Analytics," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 5, pp. 1038–1053, 2021.

[7] N. Bruschi *et al.*, "GVSoC: A Highly Configurable, Fast and Accurate Full-Platform Simulator for RISC-V based IoT Processors," in *IEEE 39th Int. Conference on Computer Design (ICCD)*, 2021, pp. 409–416.