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

TransLib, 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 fixed- and mixed-precision solutions.

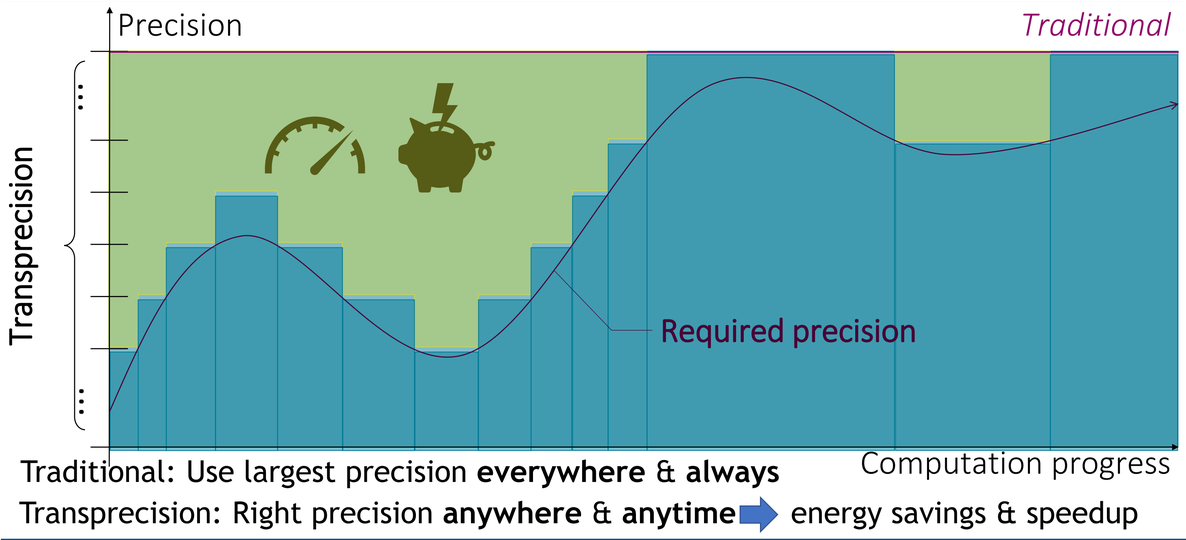
Each kernel design includes:

❖ Python model: Emulate the hardware and provide a golden reference independent of the execution of the C code

❖ C program: Including a set of optimizations portable among MCU-class targets and supporting vectorization, parallelization, fixed- and mixed-precision

https://github.com/ahmad-mirsalari/TransLib

TRANSPRECISION COMPUTING



Traditional: Use largest precision **everywhere** & **always**

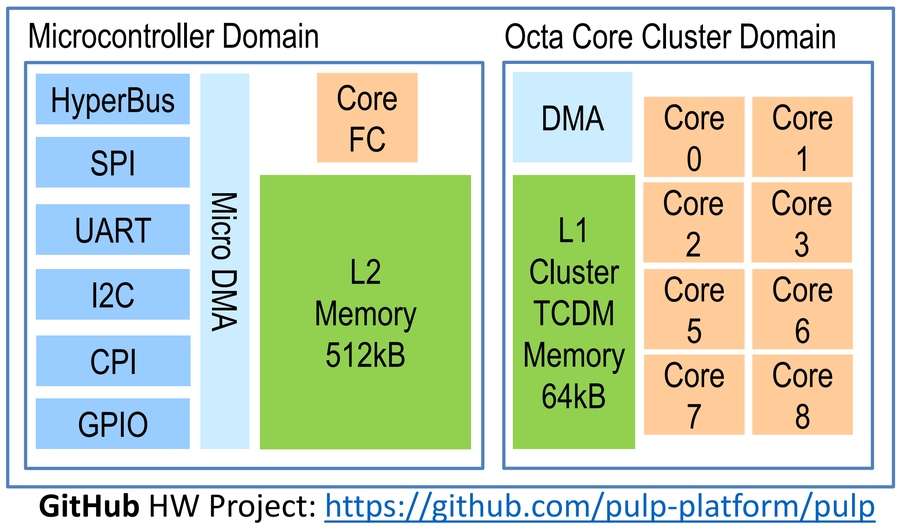
Transprecision: Right precision **anywhere** & **anytime** ➡ energy savings & speedup

CASE STUDY: PULP

- Parallel Ultra Low Power platform targeting high energy efficiencies

- A Transprecision FPU

- New low-precision data types: 32-bit , 19-bit, 16-bit, and 8-bit floating-point (FP) data types

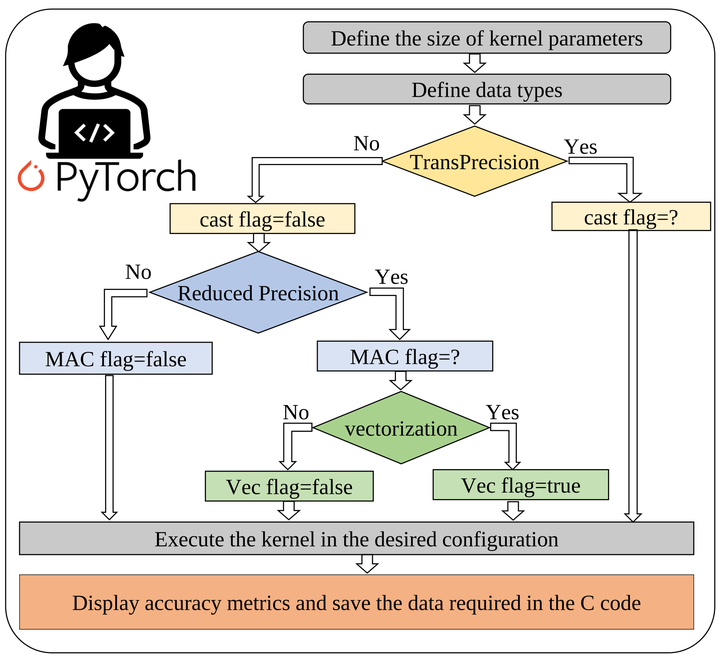


GitHub HW Project: https://github.com/pulp-platform/pulp

PYTHON MODEL

A set of flags to simulate different instructions typically available in the ISA extensions

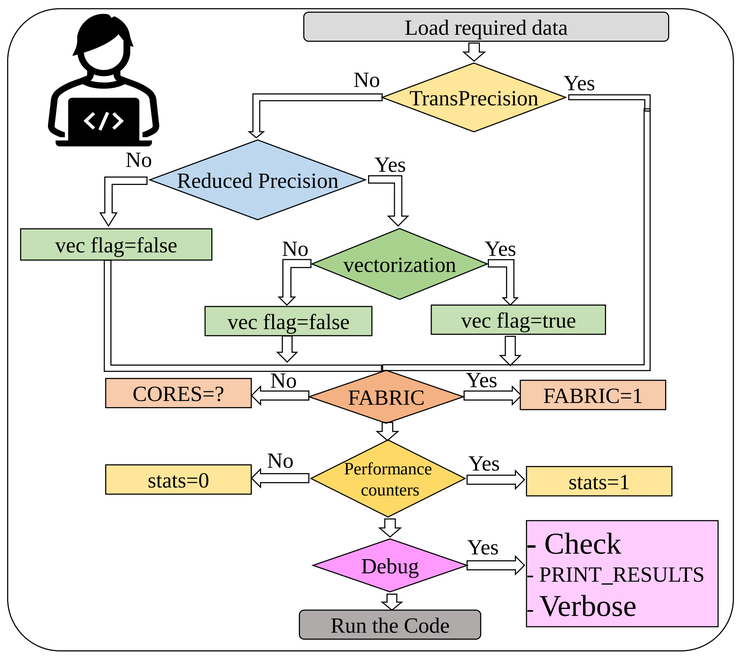
- **Cast flag**
  - Support transprecision
- **MAC flag**
  - Fused multiply-and-accumulate
- **Vector flag**
  - SIMD vector instructions
- **A set of error metrics**
  - MSE, RAE, MAE



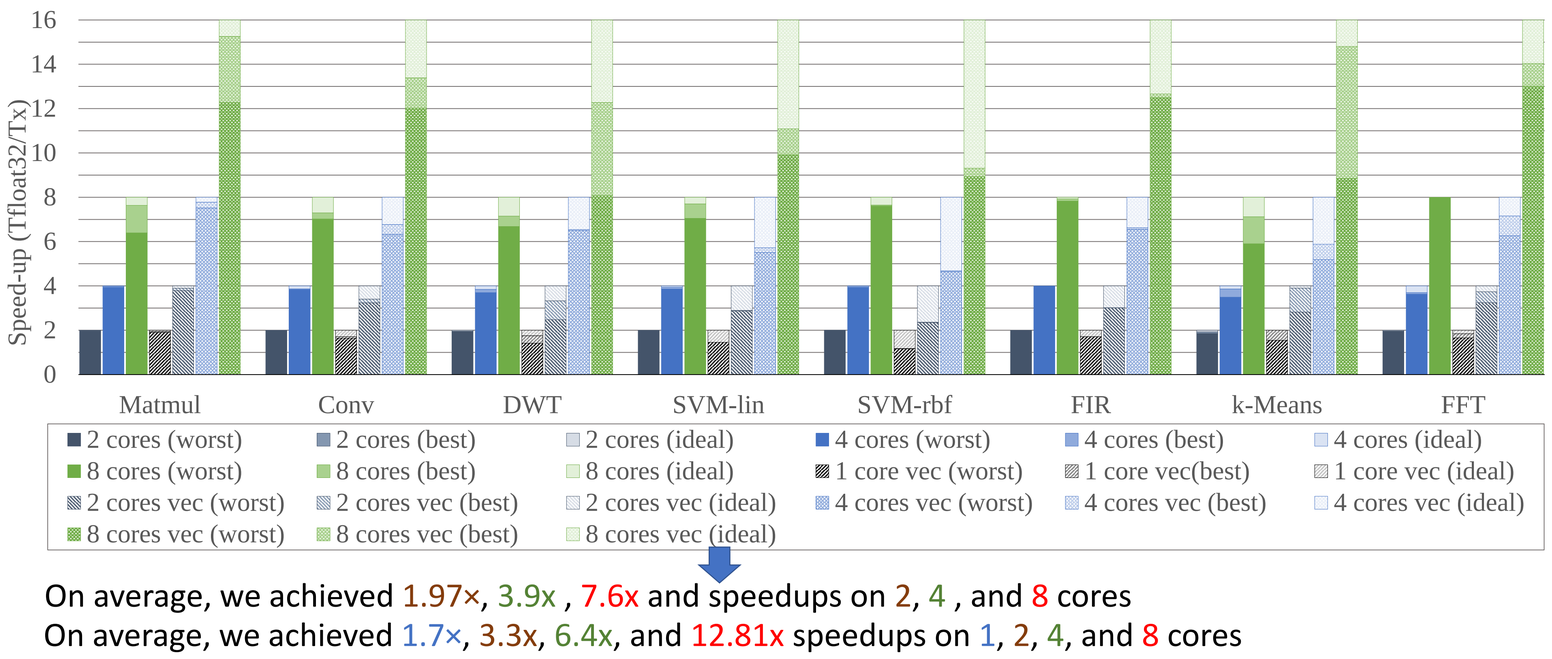
OPTIMIZED C CODE

Provide a sequential version of the C code, various parallel versions (i.e., 2, 4, and 8 cores), and all their vectorized variants

- **Vector flag**
  - SIMD sub-word parallelism
- **Parallelism**
  - Take the number of core
- **Stats flag**
  - Hardware performance counters
- **Check and debug flags**



EXPERIMENTAL RESULTS



On average, we achieved 1.97×, 3.9x , 7.6x and speedups on 2, 4 , and 8 cores

On average, we achieved 1.7×, 3.3x, 6.4x, and 12.81x speedups on 1, 2, 4, and 8 cores

Kernel	Accuracy Metrics			
	MAE	MSE	RMSE	RAE
CONV	0.0089	1.35E-04	0.0116	0.0016
MATMUL	0.0051	5.36E-05	0.0073	0.0009
FIR	0.0146	3.77E-04	0.0194	0.0013
DWT	0.0270	2.38E-03	0.0488	0.0019
KMEANS	0.0020	1.02E-05	0.0031	0.0008
FFT	0.0122	5.01E-04	0.0223	0.0049
	Classification Error			
SVM_LIN	0			
SVM_RBF	0			

Maximum error is close to zero for float16

Transprecision use cases

Kernel	MSE			Total Cycles			Memory Footprint			
	Fixed-Precision		Transprecision	Fixed-Precision		Transprecision	Fixed-Precision		Transprecision	
	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

	Compared to float32	Compared to float16
✗	Increase execution time	Increase execution time by 2.09×
✗	Increase error metrics	Memory footprint by 1.35×
+	decrease memory footprint	But the accuracy gain is around 30×

Kernel	Input Size		
	Small	Medium	Large
CONV	26.33	39.66	46.73
MATMUL	37.8	46.41	48.54
FIR	31	38.28	38.28
DWT	32.32	37.55	39.28
KMEANS	29.93	39.25	41.29
FFT	38.99	42.27	43.66
SVM_LIN	40.61	44.99	46.15
SVM_RBF	47.16	47.32	47.99

Float16 and bfloat16 reduces the memory footprint between 25% and 50%

NEXT RELEASES

❖ Expand the library by adding additional kernels

❖ Extend the support to additional FP data types (e.g., different flavors of 8-bit FP types)

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.