Accelerating AI on RISC-V Optimizing BFloat16 for Improved Efficiency

Authors

Ruhma Rizwan
Hardware Engineer
10xEngineers
Lahore, Pakistan

Umer Imran
Sr. DV Engineer
10xEngineers
Lahore, Pakistan

Maah Noor
DV Engineer
10xEngineers
Lahore, Pakistan

Zeshan Rehman
DV Engineer
10xEngineers
Lahore, Pakistan

1. Introduction

- Neural networks' are resilient/robust to numerical inaccuracies/noise.
- This has led to varied BFloat16 implementations with different degrees of compliance to IEEE-754 as can be seen in the table below.
- It allows us to balance efficiency and precision for different applications.

Feature	*RISC-V BF16 Vector Support*	*ARM's MACC Instructions*	*Intel's Deep Learning Boost*	*Google's Cloud TPUs*
IEEE-defined Rounding Modes	✓	× (RO only)	✓	× (RNE only)
Denormal Support	✓	× (Flushing)	× (Flushing)	× (Flushing)
NaN Propagation	✓	×	×	×
Exception Handling	✓	×	×	×

2. Objective

An opensource design of a possible implementation for a low-cost, modular, BFloat16 FPU micro-architecture which targets RISC-V based embedded cores.

Related literature

Neil Burgess et al. "Bfloat16 Processing for Neural Networks". In: 2019 IEEE 26th Symposium on Computer Arithmetic (ARITH). 2019, pp. 88–91. doi: 10 . 1109 / ARITH.2019.00022.

Ganesh Venkatesh, Eriko Nurvitadhi, and Debbie Marr. "Accelerating deep convolutional networks using low-precision and sparsity". In: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE. 2017, pp. 2861–2865.

3. Methodology

We have designed custom Bfloat16 instructions for a simplified implementation of IEEE-Floating point standard, suitable for low-precision tolerant applications like Neural Networks and integrated with the cv32e40p (RI5CY).

The following steps outline our approach:

- Investigation of optimization techniques.
- Development of an optimized BF16 reference model.
- Implementation of an optimized BF16 Floating Point Unit (FPU).
- Addition of custom BF16 instructions to the RI5CY core.
- Integration of the optimized BF16 FPU into the RI5CY core.
- Experimental evaluation (area, power, frequency, and accuracy measurements)

4. Results/Findings

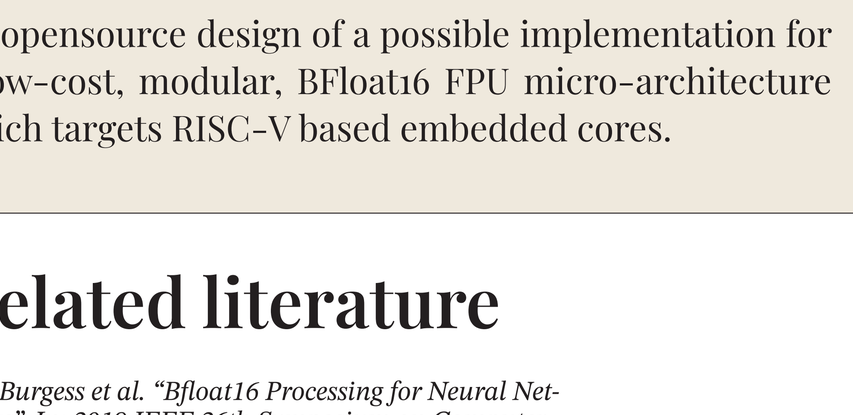
Area Savings
RTL synthesis at 65nm using Genus at 1GHz frequency

- Subnormal flushing saves area by 21~22%
- Single Rounding mode saves area by 13~14%.
- Not propagating input Nans give marginal reductions in area.

Overall reduction in area by ~35%

Polybench Benchmark

We evaluated the BF16 implementation using the PolyBench suite to measure the mean relative error of BF16 compared to FP32.

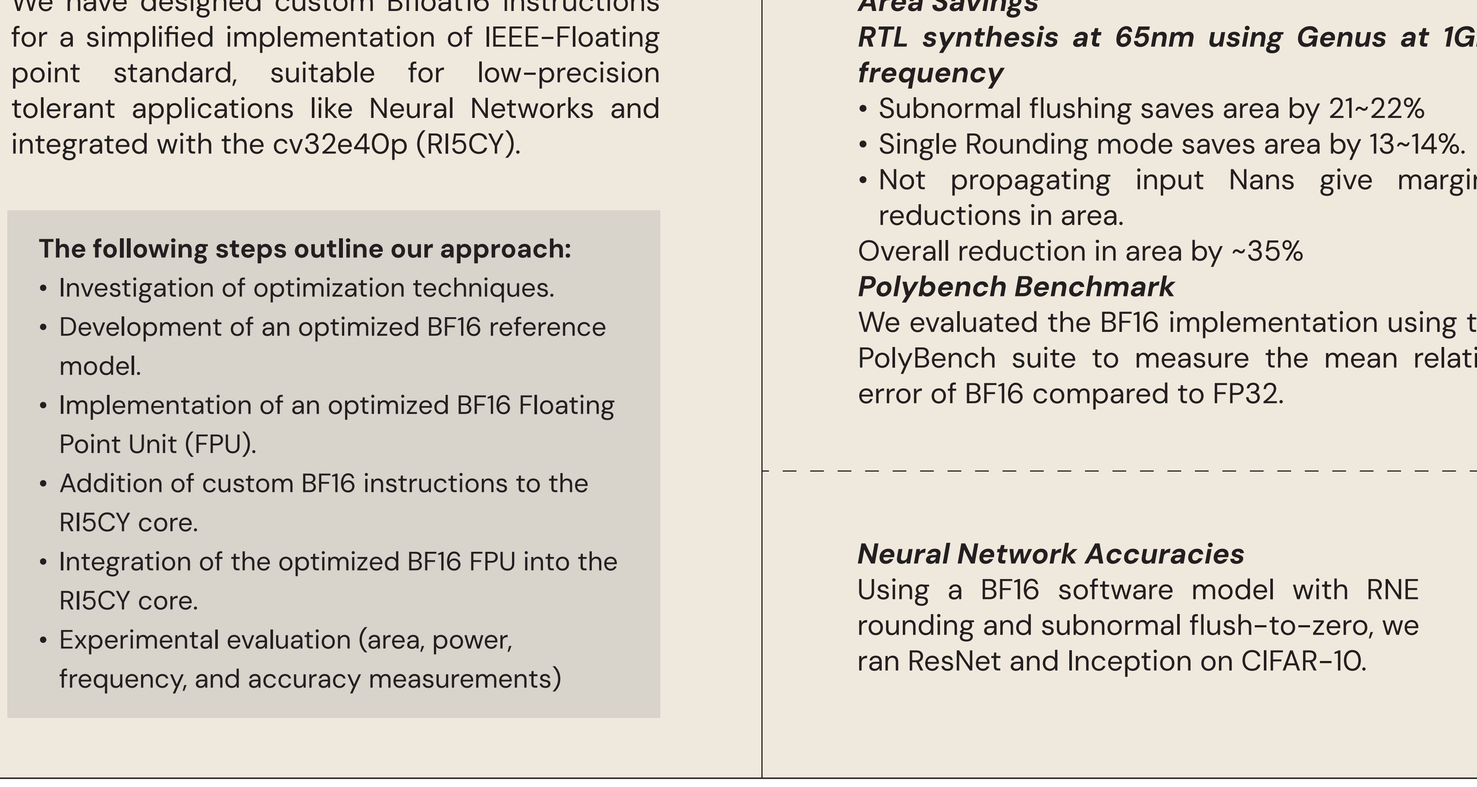


Neural Network Accuracies

Using a BF16 software model with RNE rounding and subnormal flush-to-zero, we ran ResNet and Inception on CIFAR-10.

Configuration	ResNet Accuracy	Inception Accuracy
FP32(T) / BF16(I)	0.7496	0.7754
BF16(T) / BF16(I)	0.7376	0.7243
FP32(T) / FP32(I)	0.7495	0.7761

5. Analysis



Optimizations implemented in the Optimized Bfloat16 micro-architecture

6. Conclusion

The paper proposes a power and area efficient BF16 implementation using single rounding mode, subnormal flushing, and other optimizations. Analysis shows it preserves accuracy in Polybench and NN models, enhancing embedded NN applications without significant performance trade-offs.

Get instant access to our Optimized BF16 code on GitHub – scan me!

10x ENGINEERS