TINY MACHINE LEARNING HARDWARE PLATFORM COMPUTATION OPTIMIZATION

[1]DR. CHI WAI NG, [2]MS SUK LING LI, [3]DR. PEI FUNG LAM

[1]Principal Engineer, ASTRI, Hong Kong
[2]Senior Lead Engineer, ASTRI, Hong Kong
[3]Director, ASTRI, Hong Kong
Email: [1]nevilleng@astri.org, [2]elaineli@astri.org, [3]ericlam@astri.org

Abstract - Convolution neural network (CNN) is used in many artificial intelligence (AI) applications, including voice command recognition of mobile device. Tiny Machine Learning (ML) hardware is designed to reduce computational complexity. Tiny ML implementation with block floating point (BFP) matching error tolerance of neural computing is introduced in this paper to reduce the processing and memory storage of the system. The processing time can be reduced by 50% by adopting hardware accelerators. Furthermore, the recognition results obtained from applying BFP operations are promising, achieving over 0.97recognition accuracy, with a relatively small accuracy degradation of less than 1%.

Keywords - Block floating point, Tiny Machine Learning, Convolution neural network.

I. INTRODUCTION

IoT devices are essential for smart city and smart home.However, AI applications usually require large amount ofhardware resources (computational power, memory bandwidth,etc.). Therefore, IoT devices with limited resource requiresupport from cloud back-end service. The disadvantages arehigh latency and loss of privacy. The characteristics of generalimplementation of AI system are high computation complexity,high memory bandwidth, and significant amount of sharedmemory. An innovative solution is proposed using Tiny MLplatform to solve this problem as shown in Figure 1. In thesystem, it consists of a built-in Tiny ML hardware platform toperform the complex AI calculation. For example, the tinyML platform empowered with accelerators and BFP couldrecognize voice commands to control smart city / smart homedevices such as robot cleaners and achieve 2X speed-up withnegligible accuracy degradation.

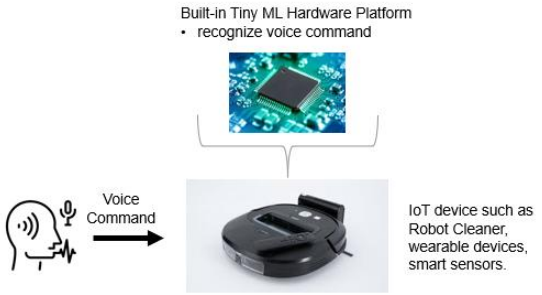


Figure 1: Tiny Machine Learning Platform

II. TINY MACHINE LEARNING PLATFORM

A Tiny Machine Learning Platform using Convolution NeuralNetwork(CNN) based on micro-processor unit (MCU) coreis proposed as shown in Figure 2.

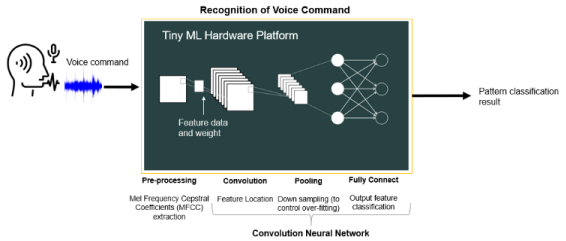


Figure 2: Convolution Neural Network

The complete CNN includes the following steps.

- Pre-processing
- Convolution
- Pooling
- Fully Connect

The “Pre-processing” extracts the features of voice commandusing Mel Frequency Cepstral Coefficient (MFCC) beforepassing to the CNN. The voice command will be streamedinto the system with at most 5 vowels. MFCC (Mel FrequencyCepstral Coefficients) is a commonly used feature extractionmethod in the field of audio processing and is widely usedin tasks such as speech recognition and music classification.The core of MFCC is to convert the spectrogram into a seriesof MFCC coefficients. Figure 3 shows the flowchartof MFCC. This process includes the following steps: MelFilter bank, Logarithmic operation, Discrete Cosine Transformand Discard low-frequency coefficients.Convolution” is thecore building blocks of a CNN. A convolution layer appliesa convolution operation to the input, passing a small filter(or kernel) over the image (or input data) to produce featuremaps. The purpose of convolution is to detect local patternsor features like edges, textures, and shapes.“Pooling” is adown-sampling operation applied after the convolutional layer.It reduces the spatial dimensions (width and height) of thefeature

map while retaining important features, performs thedown-sampling to avoid over-fit. After a series of convolutionaland pooling layers, the high-level reasoning in the neuralnetwork is done through "Fully connected layers". Theselayers are similar the layers in a traditional neural networkand are used for classification or regression tasks. The featuresextracted by the convolutional layers are flattened into a one-dimensionalvector and passed through these fully connectedlayers.

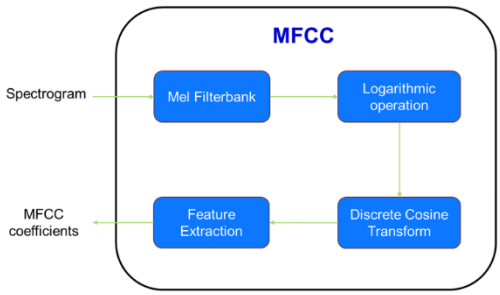


Figure 3: MFCC Flowchart

A. System Architecture

In our design, a new system architecture of Tiny MachineLearning Platform is proposed as shown in Figure 4. In thesystem, it consists of a MCU core with memory, two hardwareaccelerators including TinyML operations accelerator, SingleInstruction Multi-Data-path (SIMD) Multiply-and-Accumulate(MAC) accelerator. Tiny ML Operations Accelerator is usedto calculate the result of CNN while SIMD MAC Acceleratoris used to perform the digital signal processing such asMFCC.

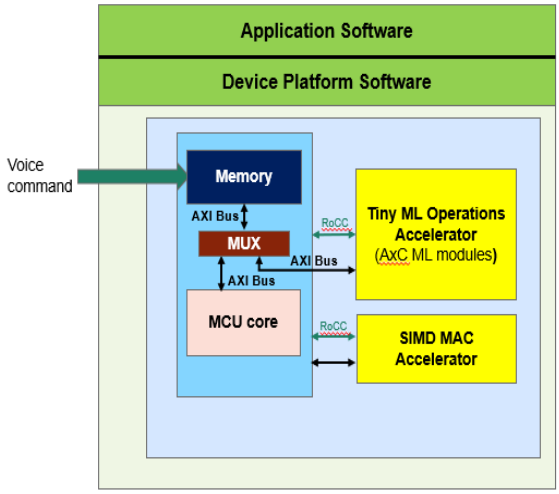


Figure 4: System architecture of Tiny Machine Learning Platform

B. Hardware Implementation

A Field Programmable Gate Array (FPGA) board is usedto implement the MCU core and hardware accelerators. TheFPGA board used is Digilent Genesys 2 as shown in Figure 5.In our design, RISC-V is used, and it is an open standardinstruction set architecture (ISA) based on established reducedinstruction set computer (RISC) principles. The RISC-V coreused is an open-source implementation from Rocket Chip,with which the RTL(Register Transfer Level) codes can becompiled using CHISEL(Constructing Hardware in a ScalaEmbedded Language) [1]. The advantage of using CHISEL isthat the configuration of the RISC-V core can be changed easily.The configuration of our RISC-V core is RV64IMAFDCand the descriptions are listed in Table 1. The RISC-V corecommunicates with the hardware accelerators using the RocketCustom Co-processor (ROCC) interface for high data transfer.

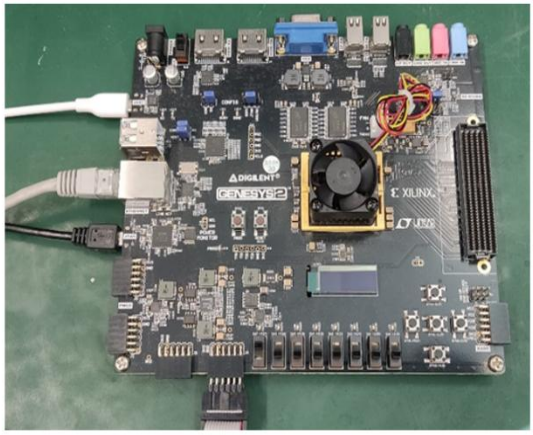


Figure 5: FPGA board

Name	Description
RV64I	Base integer instruction set, 64-bit
M	Standard extension for integer multiplication and division
A	Standard extension for atomic instructions
F	Stand extension for single-precision floating point
D	Standard extension for double-precision floating-point
C	Standard extension for compressed instructions

Table 1 ISA BASE AND EXTENSIONS OF RISC-V

C. Software Implementation

TFLite Micro from Tensorflow is an infrastructure to enabledeployment of ML models to low-power resource-constrainedembedded targets (including microcontrollers and digital signalprocessors). Originally model size is target to less than20KB, which can fit within 32KB of read-only(flash) memory.This doesn't produce particularly accurate results. The trainedmodel must convert the input and output values from float tointeger, which is called quantitation.

In our design, the input data, the intermediate algorithmiclogics, and the output data are modified such that the trainedmodel in float can be directly used on RISC-V. No quantitationis needed anymore. Hence, the interference accuracy can alsobe improved. Figure 6 and figure 7 show the trained models with and without quantization.

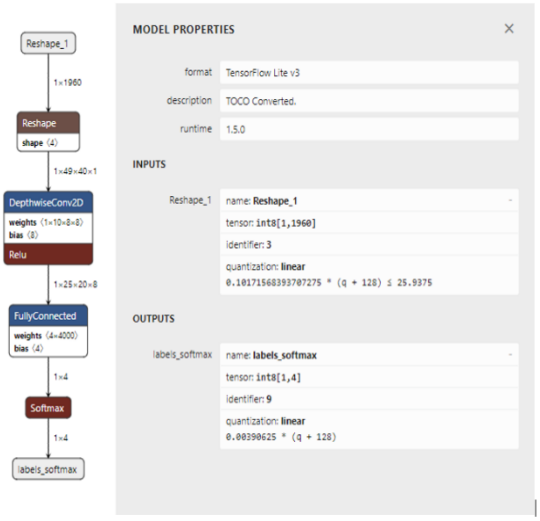


Figure 6: Trained Model with Quantization (Integer)

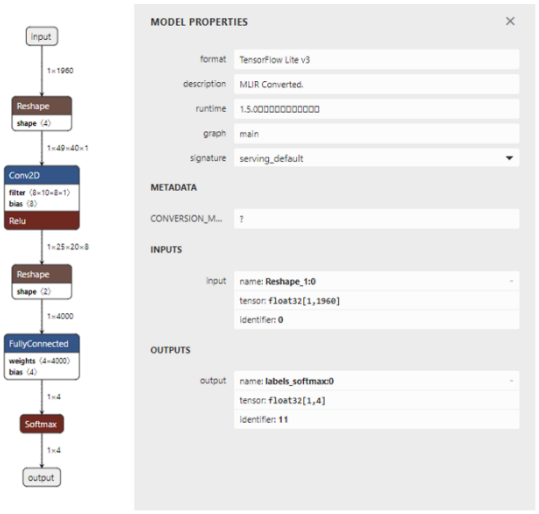


Figure 7: Trained Model without Quantization (Float)

The TensorFlow Lite model doesn't take in raw audiosample data. Instead, it works with spectrograms, which aretwo dimensional arrays that are made up of slices of frequencyinformation, each taken from a different time window. Figure8 shows the flow chart of the inference process. Theinput audio data is first converted into spectrogram, and thenprocessed by the trained model. If there is a voice commandbeing recognized, a string and an accuracy of the recognizedvoice command will be output.

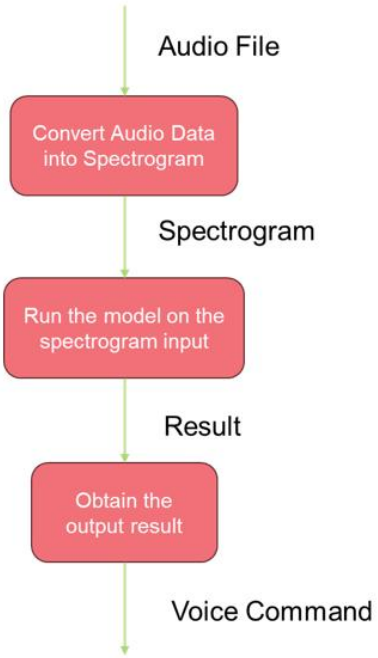


Figure 8: Flow Chart of Inference Process

III. CONVOLUTION USING BLOCK FLOATING POINT (BFP)

The diagram Figure9 shows the software flow chartof main arithmetic operation in the convolution layer fromTensorflow tflite-micro when basic data type FP32 is convertedto BFP. The flow is split into three stages: Data preparation,BFP operations and operations, and Output [2] [3].

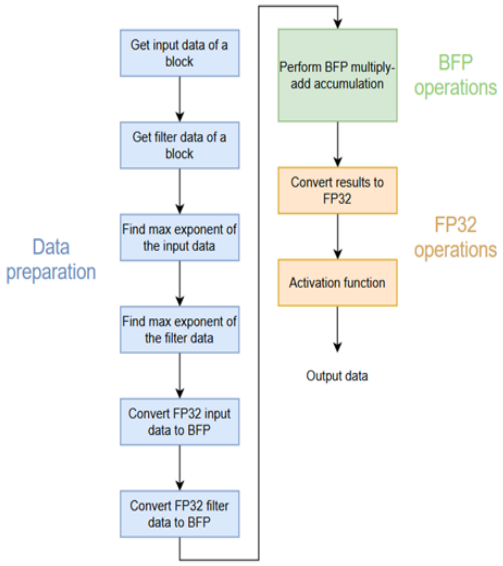


Figure 9: Software Flowchart of CNN

Most of the processing time is spent on the Data preparationand BFP arithmetic operations. Fortunately, these blocks canbe optimized by using hardware accelerators to reduce theprocessing time. The data path of the block floating point isshown in Figure10.

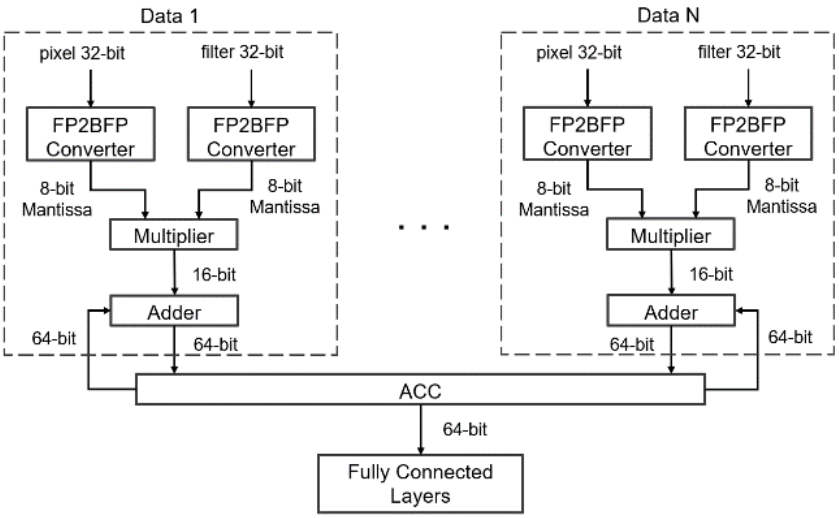


Figure 10: Datapath of Block Floating Point

In our design, 8-bit x 8-bit multiplier is used, and theresults will be accumulated into a 64-bit accumulator to avoidover-flow. After the completion of the operation, the blockfloating point will not be converted to floating point and keptunchanged for further processing [4] [5] [6] [7].

In the Data preparation stage, the operations can be optimizedby parallel processing, essentially halving the processingtime required. Each conversion from FP32 to BFP datacan be performed right before a BFP arithmetic operation tosave the processing time. In the BFP operations stage, theexponent part of floating point is fixed to the maximum valuealong a block of input data. The mantissa part is roundedofffrom 24 bits to 8 bits from MSB, then the intermediateresult of the multiplication and addition of a block of data isstored. By removing several repeated shift operations in eachmultiplication and addition, the processing time can be roughlyreduced by 50%. Since the mantissa part is reduced from 24-bit to 8-bit, the memory storage is also roughly reduced by50%.

The output data in BFP format is then passed to thenext stage – "Fully Connected" for the decision makingof classification or regression tasks, which flattens extractedfeatures into a one-dimensional vector. Figure 11shows the flow chart of each stage when they are optimizedby hardware accelerator. In the Data preparation stage, themaximum exponent value along a pair of input block datacan be first computed and stored up for BFP operations touse in next loop. By parallelization of Data preparation and BFP operations, the processing time of Data preparation stagecan be cancelled out. Since the flow is to be repeated many times until the whole convolution layer iscompleted, the three stages can be pipelined in hardware forthe speed enhancement.

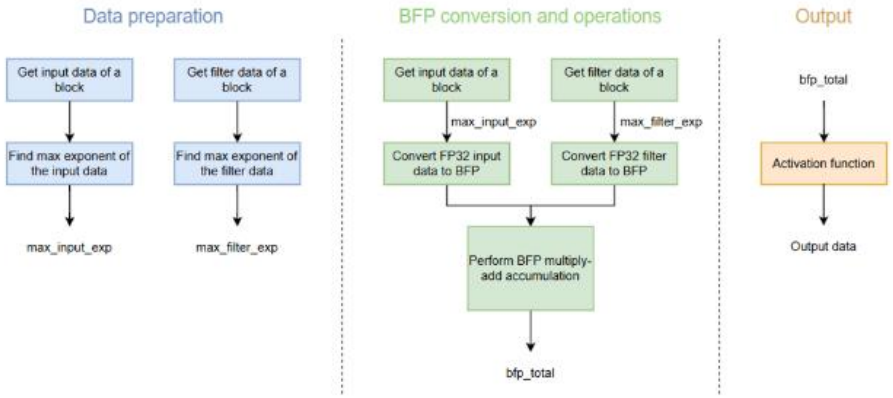


Figure 11: Convolution using Block Floating Point

A. Comparison with other Block Floating Point

An algorithm of Fast First, Accurate Second Training(FAST) system for DNNs(Deep Neural Networks) is introducedin [8]. This paper focuses on optimizing the trainingtime while our design target is for the inference process.The disadvantage of variable precision BFP is that it requiresmultiple passes of the FAST Multiplier-Accumulator (fMAC)while our design requires only one pass of MAC. The loworderbits of mantissa part are

truncated while our design willkeep all mantissa bits in MAC operation. BFP with stochasticrounding was used. This may lead to the accuracy distortion.Our design does not need to apply the stochastic rounding forthe truncation of low-order bits of mantissa part.
A hybrid block floating point approach is introduced in [9].All dot-product operations (i.e., convolutions, matrix multiplications,and outer products) in DNNs will be processed in BFPformat, while other operations (i.e., activations, regularizations,etc.) will be performed in floating point. This approachrequires many BFP-to-FP conversions which may negativelyimpact the overall performance. In our design, BFP will beused in all arithmetic of the convolution layer. The outputwill then be passed to the "Fully Connected" layer for thedecision making. This design saves the processing overheadin BFP-to-FP conversions by adopting 64-bit width mantissaaccumulator which is used to store the multiplication result to avoid divergence.

IV. EXPERIMENTAL RESULTS

To evaluate the effect on accuracy after FP32 operations are replaced by BFP, micro speech application which can recognize voice command in tflite-micro, is run on an FPGA which implemented the RISC-V core with floating point unit. 4 different length voice commands are used for testing. Therecognition results are summarized in table 2. All 4 commandsare recognized correctly on FPGA Board. The accuracy of usingfloating point and BFP are all over 0.97 and 0.97, respectively.Most importantly, the accuracy degradation is only below 1%for all 4-test data by applying BFP operations. The voicecommand is read from a file as an input data and fed intoTFLite Micro infrastructure to recognize the command. Therecognition result will be displayed on PC through theUARTinterface as shown in Figure 11.

Command	Float Accuracy	BFP Accuracy	Accuracy degradation
Backward	0.99852720	0.99835590	0.0172%
Turn Right	0.99948280	0.99943455	0.0048%
Forward	0.98652060	0.98601625	0.0511%
Turn Left	0.97706765	0.97513085	0.1982%

Table 2 RECOGNITION RESULT

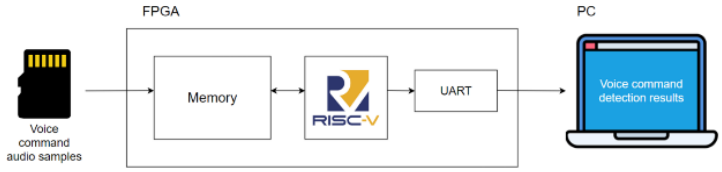


Figure 11: Recognition flow diagram

ACKNOWLEDGMENT

The research work of this paper is supported by CNERC.

V. CONCLUSION

A tiny machine learning hardware platform for voice commandrecognition is proposed in this paper. By designing hardwareaccelerators with BFP arithmetic operations, the computationalcomplexity is significantly reduced, and the processingtime can be reduced by approximately 50%. Experimentalresults show that the accuracy of using BFP is consistentlyabove 0.97, and the accuracy degradation is negligible at below1%.

REFERENCES

[1] Martin Schoeberl, Digital Design with Chisel, Fifth Edition 2019

[2] Xiaocong Lian ,Zhenyu Liu, Zhourui Song, Jiwu Dai, Wei Zhou, andXiangyang Ji, "High-Performance FPGA-Based CNN Accelerator WithBlock-Floating-Point Arithmetic,"IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, VOL. 27, NO. 8,pp.1874 - 1885. AUGUST 2019.

[3] David Elam and Cesar Iovescu, A Block Floating Point Implementation for an N-Point FFT on the TMS320C55x DSP, Application Report,September 2003.

[4] Daniel Lo, Amar Phanishayee, Eric S. Chung, Yiren Zhao, Ritchie Zhao,Neural Network Activation compression with Outlier Block Floating-Point, US Patent, December 2018.

[5] Daniel Lo, Amar Phanishayee, Eric S. Chung, Yiren Zhao, Ritchie Zhao,Neural Network Activation compression with Narrow Block Floating-Point, US Patent, December 2018.

[6] Douglas C. burger, Eric S. Chung, Bita Darvish Rouhani, Daniel Lo,Ritchie Zhao, Design Flow For Quantized Neural Networks, US Patent,May 2018.

[7] Bita Darvish Rouhani, Eric S. Chung, Daniel Lo, Douglas C. Burger,Adjust Precision and Topology Parameters For Neural Network Training Based on a Performance Metric, US Patent,Dec 2018.

[8] Sai Qian Zhang, Bradley McDanel, and H.T. Kung, FAST: DNNTraining Under Variable Precision Block Floating Point with StochasticRounding, 28th IEEE Int Symp on High-

Performance Computer Architecture(HPCA-28), February 2022.

[9] Mario Drumond, Tao Lin, Martin Jaggi, Babak Falsafi, Training DNNswith Hybrid Block Floating Point, 32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montr´eal, Canada.

★★★