Real-Time Keyword Spotting: An FPGA-Based Inference Accelerator

Harry Minksy (hm3121), Opalina Khanna (ok2373), Sunny Fang (yf2610), Aaron Zhu (azz2111)

March 13, 2026

1 Introduction

At its core, this project aims to understand the advantages and drawbacks of running inference using a quantized Convolutional Neural Network (CNN) on an FPGA, and utilizing its inherent parallelism.

Broad function of project: Trained, quantized Speech Recognition model running on the FPGA, processing audio input from the audio on the board, rendering the recognized speech as a display on the VGA. The model will be trained offline on a workstation using PyTorch and quantized to INT8 (or possibly INT4) before deployment. The Verilog aspect of the system implements a computationally intensive DSP front-end and the convolutional layer. The C software handles weight loading from the SD card, and the lightweight fully-connected classification head, softmax, and label smoothing over time.

2 Components

Audio coder/decoder (CODEC) of the DE1-SoC allows for audio processing and interfacing with microphones and speakers, providing 48000 samples per second (48 kHz) by default. The audio CODEC collects raw audio samples [2]. Mel Frequency Cepstral Coefficients (MFCC) are used as a numerical representation that can be used as an input.

SD card interface will store the model weight files resulting from quantization, which is how the model will be loaded into the DE1-SoC board for inference. There are two possible routes.

The ONNX format can be exported from a PyTorch quantized model and executed on the HPS as a software process, with weights loaded directly into DDR3 RAM via standard Linux file I/O.

Alternately, FINN compiles Brevitas-quantized models into dataflow-style architectures, where weights are stored in the FPGA's on-chip BRAM and accessed through the HPS-to-FPGA bridge via the lightweight AXI bridge interface on the DE1-SoC.

A current sensor (TBD) such as INA169, INA 219, similar will be used to monitor the power of the whole board. It outputs an analog voltage proportional to current [1]. It may be challenging to isolate energy consumption of model inference, which we plan to remediate by establishing a baseline with the energy consumption measured when the board is idle.

Framebuffer will display recognized speech.

3 Major Tasks

3.1 Model Training

- ML Model for Speech Recognition: We will first train a Keyword Spotter (KWS) with PyTorch, using this tutorial as a reference. We are considering using the speech commands dataset collected by Google or a similar audio dataset [5].
- Quantization: A choice between the following
  - Post-training Quantization (PTQ): With no training required, 8-bit quantization can be done within PyTorch.
  - Quantization-aware training (QAT): With brevitas, we can also conduct QAT for 8-bit and 4-bit precision [4]. After training and quantization, we will export the weight files.

3.2 Audio Signal Processing

- At a high level, we will process a vocal input using a microphone plugged into the via the MIC IN on the boards audio CODEC. Based on similar projects we've seen[3], an ideal pipeline for processing the audio would be to first reduce noise using an FIR filter, convert the signal to the frequency domain using the Fast Fourier Transform, and perform feature extraction using a method such as identifying the Mel-Frequency Cepstral Coefficients. Given that our experience in digital signal processing is limited, we think that a reasonable scope for this project is the implement the FIR filter and feature extraction, and to use existing IP cores for the FFT (https://www.altera.com/products/ip/po-3074/fft-fpga-ip-cores). Once the audio signal has been pre-procesed into feature vectors, it will be pipelined into our classification module.

3.3 Running Inference

3.3.1 Software Details

- Weight Loading: At startup, the driver reads the quantized INT8 file from the SD card and writes each weight value into the memory mapped weights register bank on the FPGA via the HPS-to-FPGA lightweight bridge. This requires correct address mapping and proper use of volatile pointers.
- FC Classification: After each conv layer pass, the driver reads the activation map from the FPGA-side BRAM, flattens it, and runs the fully connected layer (matrix-vector multiply) and softmax.
- Label Smoothing: Raw per-frame softmax outputs are noisy. The driver maintains a sliding window of the last 5 frame predictions and averages the softmax distributions before taking the argmax, to significantly reduce the flicker in output.

3.3.2 Hardware Details

- DSP Front-End (as described above)
- Convolutional Engine: The convolution engine slides a 3x3 window across the input feature map, extracting nine INT8 values per step. Eight parallel MAC units each compute the dot product of those values against one stored filter kernel simultaneously, and a depth-4 adder tree accumulates the partial products into a single output activation per filter. Kernel weights are loaded into on-chip registers by the C driver at startup and remain fixed for the duration of inference.

References

[1] Power estimation/measurement in quartus prime cornell ece5760. URL https://people.ece.cornell.edu/land/courses/ece5760/DE1_SOC/HPS_peripherials/Power_est_index.html.

[2] Audio core, 2026. URL https://www-ug.eecg.toronto.edu/msl/nios_devices/dev_audio.html.

[3] E. Arellanos, L. D. Remigio, J. L. Ostos Marquez, and M. Nunez. Voice recognition acquisition and storage system using vhdl and the de1-soc fpga. In *2024 IEEE XXXI International Conference on Electronics, Electrical Engineering and Computing (INTERCON)*, pages 1–8, Nov 2024. doi: 10.1109/INTERCON63140.2024.10833462.

[4] G. Franco, A. Pappalardo, and N. J. Fraser. Xilinx/brevitas, 2025. URL https://doi.org/10.5281/zenodo.3333552.

[5] P. Warden. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. *ArXiv e-prints*, Apr. 2018. URL https://arxiv.org/abs/1804.03209.