MNIST Neural Network Inference Accelerator

Ifesi Onubogu (io2249), Colin Paul Jaworowski (cpj2118)

Spring 2026

Project proposal for CSEE 4840 Embedded System Design

1 Overview

The core of this project is a hardware accelerator for neural network inference on the DE1-SoC FPGA platform. The accelerator will classify handwritten digit images (28×28 pixels) from the MNIST dataset using a small, pre-trained multi-layer perceptron (MLP) network. The project serves as a simplified model of modern AI inference accelerators, demonstrating the key concepts of quantization, hardware multiply-accumulate (MAC) operations, and hardware-software co-design.

Software running on the ARM Cortex-A9 Hard Processor System (HPS) will handle loading pre-trained model weights, preprocessing input images, and managing user interaction. The FPGA fabric will contain the inference accelerator: a MAC array that performs the matrix-vector multiplications at the heart of neural network inference. The result—a predicted digit (0–9)—will be displayed on the 7-segment displays and optionally on a VGA-connected monitor along with the input image.

The trained model will be a two-layer MLP ($784 \rightarrow 128 \rightarrow 10$) achieving $\sim$97% accuracy on MNIST, trained in PyTorch on a host PC and quantized to 8-bit integers for efficient hardware implementation.

2 System Architecture

Figure 1 shows the overall system architecture. The HPS communicates with the FPGA accelerator over the Avalon memory-mapped lightweight bridge. Weights and image data flow from the HPS into on-chip BRAM, while inference results flow back to the HPS and to the display outputs.

3 Technical Details

3.1 Neural Network Architecture

The network is a simple two-layer multi-layer perceptron (MLP):

Layer	**Input**	**Output**	**Parameters**
Fully Connected 1	784	128	100,352 weights + 128 biases
ReLU Activation	128	128	0 (combinational)
Fully Connected 2	128	10	1,280 weights + 10 biases
Argmax	10	1	0 (combinational)

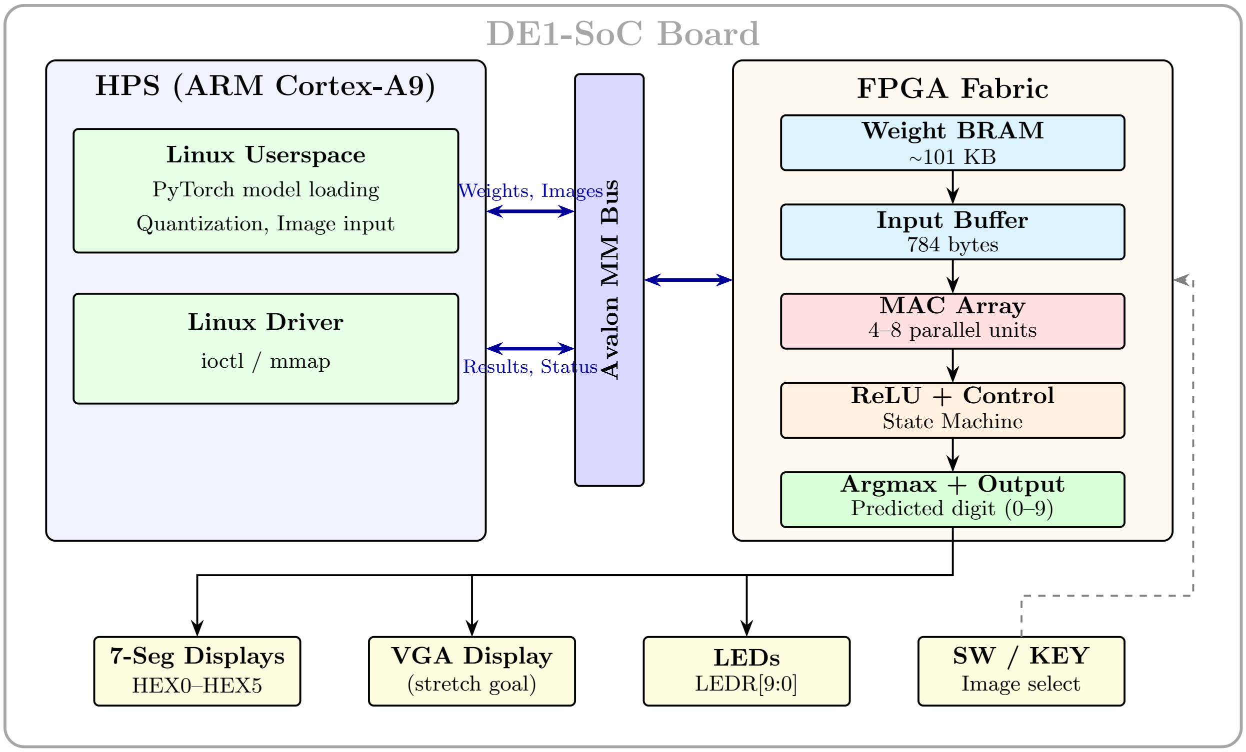

Figure 1: System block diagram showing HPS–FPGA communication via the Avalon memory-mapped bus, the inference accelerator pipeline in the FPGA fabric, and output peripherals.

Total parameters: 101,770
Storage at 8-bit quantization: ~101 KB (fits in Cyclone V on-chip BRAM)
Expected accuracy: ~97% on MNIST test set

This architecture was chosen because it is small enough to fit entirely in on-chip BRAM (the Cyclone V 5CSEMA5F31C6 has ~557 KB of embedded memory), avoiding the complexity and latency of external DRAM access during inference.

3.2 Quantization

The model will be trained in PyTorch using full 32-bit floating point, then quantized to 8-bit integers using PyTorch's built-in post-training quantization (`torch.quantization`). This is standard practice in edge AI deployment and reduces:

- **Memory:** 4× reduction (32-bit → 8-bit)
- **Compute:** Integer MAC operations instead of floating-point, which are far more efficient on FPGA
- **Accuracy impact:** Typically <1% accuracy loss for this simple network

Weights and activations will both be quantized to signed 8-bit integers (INT8). The accumulator will use 32-bit integers to prevent overflow during MAC operations.

3.3 Hardware Accelerator Design

The accelerator implements a systolic-style MAC unit that computes one layer of the network at a time. For each output neuron j, the operation is:

```
accumulator = bias[j]
for i = 0 to N-1:
    accumulator += weight[j][i] * input[i]
output[j] = ReLU(accumulator)    // max(0, accumulator)
```

Key hardware components:

- **Weight BRAM:** Stores all quantized weights and biases. Organized as multiple BRAM blocks. Weights are loaded once at startup via the Avalon bus from the HPS.
- **Input Buffer:** A 784-byte BRAM that holds the current input image (28×28 pixels, 8-bit grayscale).
- **MAC Array:** One or more multiply-accumulate units ($8-bit \times 8-bit \rightarrow 32-bit$ accumulate). Using multiple MAC units in parallel allows computing multiple output neurons simultaneously. Target: 4–8 parallel MACs.
- **ReLU Unit:** Simple comparator—if accumulator < 0, output 0; otherwise, pass through. Purely combinational logic.
- **Argmax Unit:** Finds the index (0–9) of the largest value among the 10 output neurons. This gives the predicted digit.
- **Control State Machine:** Sequences the computation through Layer 1, ReLU, Layer 2, and Argmax. Signals completion to the HPS via a status register.

Estimated inference latency with 4 parallel MACs at 50 MHz:

- Layer 1: $784 \times 128/4 = 25{,}088$ cycles $\rightarrow \sim$0.5 ms
- Layer 2: $128 \times 10/4 = 320$ cycles $\rightarrow \sim$0.006 ms
- **Total: <1 ms per image** (vs. ∼5–10 ms in pure software on the ARM core)

3.4 Hardware-Software Interface

Communication between the HPS and FPGA accelerator will use the **Avalon memory-mapped (MM) lightweight bridge**, which maps FPGA registers into the HPS address space.
Operation sequence:

1. At startup, HPS loads quantized weights into FPGA BRAM via `WEIGHT_ADDR/WEIGHT_DATA` registers.
2. For each inference: HPS writes 784 pixels via `IMAGE_ADDR/IMAGE_DATA`, then asserts `START` in `CONTROL`.
3. Accelerator computes through both layers, writes result to `RESULT` register, sets `STATUS.done`.
4. HPS reads `RESULT` to get the predicted digit (0–9).

Offset	Name	R/W	Description
`0x00`	`CONTROL`	W	Bit 0: start inference, Bit 1: load weights
`0x04`	`STATUS`	R	Bit 0: busy, Bit 1: done
`0x08`	`RESULT`	R	Predicted digit (0–9)
`0x0C`	`CONFIDENCE`	R	Output value of winning neuron
`0x10`	`WEIGHT_ADDR`	W	Address for weight loading
`0x14`	`WEIGHT_DATA`	W	Data for weight loading
`0x18`	`IMAGE_ADDR`	W	Address for image pixel loading
`0x1C`	`IMAGE_DATA`	W	8-bit pixel data

Table 1: Accelerator register map.

3.5 Display Output

- **7-Segment Displays (HEX0–HEX5):** HEX0 shows the predicted digit (0–9). HEX2–HEX1 show the confidence score. HEX5–HEX3 show the image index.
- **VGA Display (optional stretch goal):** Render the 28×28 input image scaled up ($10\times$ to 280×280) alongside the prediction result.
- **LEDs:** LEDR[9:0] display a one-hot encoding of the predicted digit.

3.6 Input Method

The user will select test images using the **switches (SW[9:0])** to choose an image index (0–1023) from a preloaded test set stored in DDR3 memory on the HPS side. Pressing **KEY[0]** triggers inference on the selected image. This provides a simple, self-contained demo without requiring external peripherals.

4 Software Components

4.1 PyTorch Training Script (Host PC)

- Train a $784 \rightarrow 128 \rightarrow 10$ MLP on MNIST using PyTorch
- Apply post-training INT8 quantization
- Export quantized weights and biases as binary files

4.2 HPS Application (ARM Linux)

- Load quantized weight files from SD card
- Transfer weights to FPGA BRAM at startup
- Read user input (image selection via switches)
- Transfer selected test image to FPGA and trigger inference
- Read result and update displays

4.3 Linux Device Driver

- Kernel module mapping the Avalon lightweight bridge
- Provides `ioctl` interface for: loading weights, sending images, starting inference, reading results
- Optional: interrupt-driven notification when inference completes

5 Major Tasks

- **Design decisions** for all major points (MAC parallelism, BRAM organization, Avalon interface details, quantization strategy). These will be in the design document.
- **PyTorch training and quantization pipeline** on a host PC, producing exportable INT8 weight files and a reference software implementation for accuracy verification.
- **Verilator-based testbench** able to run the hardware accelerator in simulation to verify MAC operations, layer computation, ReLU activation, and argmax output. The testbench will compare hardware results against the PyTorch reference for a set of test images.
- **Hardware accelerator source code** in SystemVerilog, consisting of: weight and input BRAM modules, MAC array with configurable parallelism, ReLU activation (combinational), argmax output selection, control state machine, and Avalon MM slave interface.
- **Linux device driver** for the hardware accelerator with a high-level C interface for loading weights, sending images, and reading inference results.
- **HPS userspace application** that loads the trained model, handles user input, orchestrates inference, and displays results on 7-segment displays (and optionally VGA).
- **Accuracy validation** comparing hardware inference results against PyTorch software results across the full MNIST test set (10,000 images) to verify quantization fidelity.

6 Feasibility Summary

The table below demonstrates that this project is well within the resource constraints of the DE1-SoC platform:

Resource	**Required**	**Available (Cyclone V)**	**Utilization**
Weight Storage	∼101 KB	∼557 KB BRAM	∼18%
Logic Elements	∼5K–10K (est.)	∼85K	∼6–12%
Clock	50 MHz	50 MHz	—
Inference Time	<1 ms	—	—
DSP Blocks	4–8 (for MACs)	87 available	∼5–9%

Table 2: FPGA resource utilization estimates.

The project uses a small fraction of available FPGA resources, leaving ample room for iteration and potential enhancements such as additional MAC parallelism or VGA display output.