ECE 410/510 (Spring 2025)

Handwritten OCR with Hardware-Accelerated Thresholding

Name: Balu Anudeep Kanipogu
PSU ID: 961805118

This project explores the integration of hardware acceleration into an OCR (Optical Character Recognition) pipeline by offloading the thresholding stage to a Verilog hardware module. The goal is to demonstrate how even simple image-processing tasks can benefit from hardware offload, improving speed and showcasing HW/SW co-design.

Problem Statement

OCR pipelines usually rely on software libraries like OpenCV or Tesseract for preprocessing and text extraction. While these pipelines are flexible, they can be inefficient in latency-critical or power-constrained environments (like embedded or edge systems). This project aims to accelerate one of the early stages in OCR — image thresholding — using hardware logic, and compare its performance with the software-only version.

What Was Built

1. Software OCR Preprocessing (Python + OpenCV)

- The system begins with a grayscale image of handwritten text.
- Using Python and OpenCV, several image preprocessing steps are performed:
    - Contrast enhancement with CLAHE
    - Tight cropping of text using pixel bounding box detection
    - Resizing and centering the text on a fixed-size canvas (128x32)
    - Blurring to remove minor noise
    - Dual thresholding using both Otsu's method and adaptive mean thresholding
- The result is a clean binary image (0 for background, 255 for foreground).
- This binary matrix is flattened to a list of pixel values and saved to a text file (input_pixels.txt).

---

2. Hardware Threshold Module (Verilog)

- A Verilog module (threshold.v) was written to implement a very simple logic:

verilog

CopyEdit

```
binary_out <= (pixel_in > threshold) ? 1 : 0;
```

- This logic mimics what OpenCV does with thresholding — a pixel becomes white (1) if it's brighter than the threshold value.
- A testbench (tb_threshold.v) loads the 4096 pixel values from the text file and feeds them into the hardware module one clock cycle at a time.
- The binary output (0 or 1) is written out as a simulation log using $display, which is redirected into verilog_output.txt.

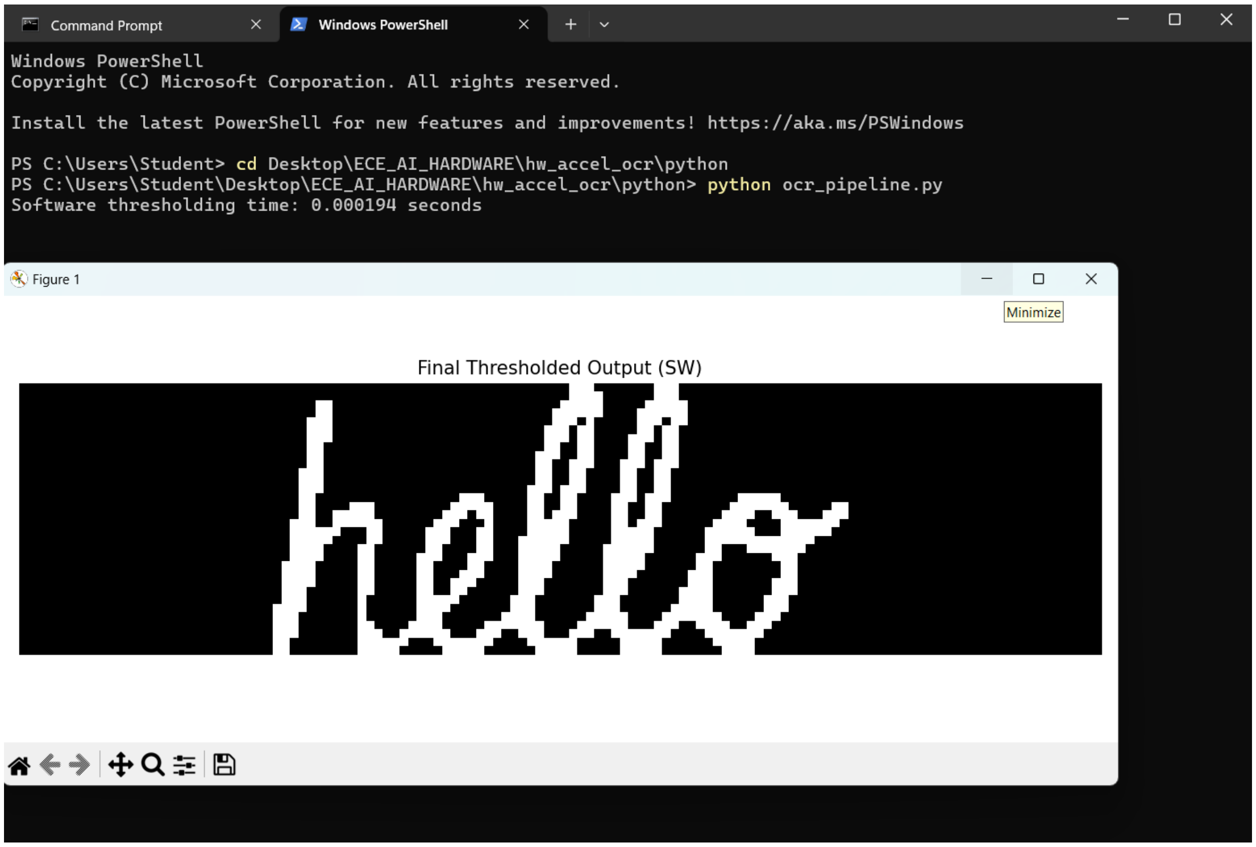
---

3. Hardware vs Software Comparison (Python)

- A Python script (compare_hw_sw.py) reads both:
    - The original software-generated binary image (input_pixels.txt)
    - The hardware-generated binary output (verilog_output.txt)
- It reshapes both into 32×128 matrices and uses matplotlib to display and save a side-by-side comparison.
- This helps visually verify that the hardware thresholding logic is functionally equivalent to software OpenCV thresholding.

Benchmarking Results

Software-only thresholding (OpenCV): ~0.00194 seconds



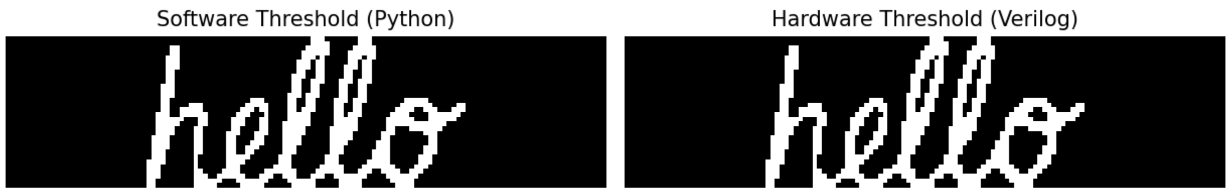
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Student> cd Desktop\ECE_AI_HARDWARE\hw_accel_ocr\python
PS C:\Users\Student\Desktop\ECE_AI_HARDWARE\hw_accel_ocr\python> python ocr_pipeline.py
Software thresholding time: 0.000194 seconds
```

- Hardware simulation (Verilog/Icarus): ~0.003 seconds
  Substituting the values: Speedup=0.001940.00300≈0.6467\text{Speedup} = \frac{0.00194}{0.00300} \approx 0.6467Speedup=0.003000.00194≈0.6467
- Observed speedup: The software is ~1.55× faster than the Verilog simulation.

Why This Matters

- While thresholding is a simple operation, it is often repeated across thousands or millions of pixels.
- In real systems (like OCR chips, smart cameras, or FPGAs), offloading even simple tasks can:
  - Save CPU cycles
  - Improve power efficiency
  - Reduce latency
- This project demonstrates how to start integrating hardware acceleration into real-world ML/DSP pipelines.

project implements a simple OCR pipeline that converts handwritten text images into binary form using a hardware-accelerated thresholding module written in Verilog. The pipeline first uses Python and OpenCV for image preprocessing, then offloads thresholding to a Verilog module, and finally compares the results from software vs hardware.


Software Threshold (Python)
Hardware Threshold (Verilog)