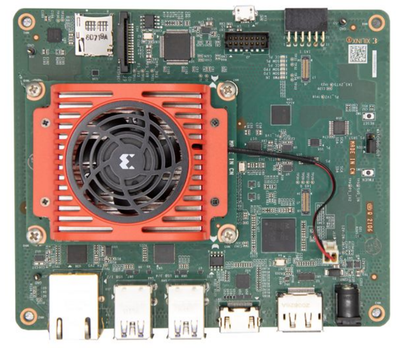
Texelerate

E1: Andrew Liao, Amelia Heller, Anirudh Prakash
18-500 Capstone Design, Spring 2025
Electrical and Computer Engineering Department
Carnegie Mellon University



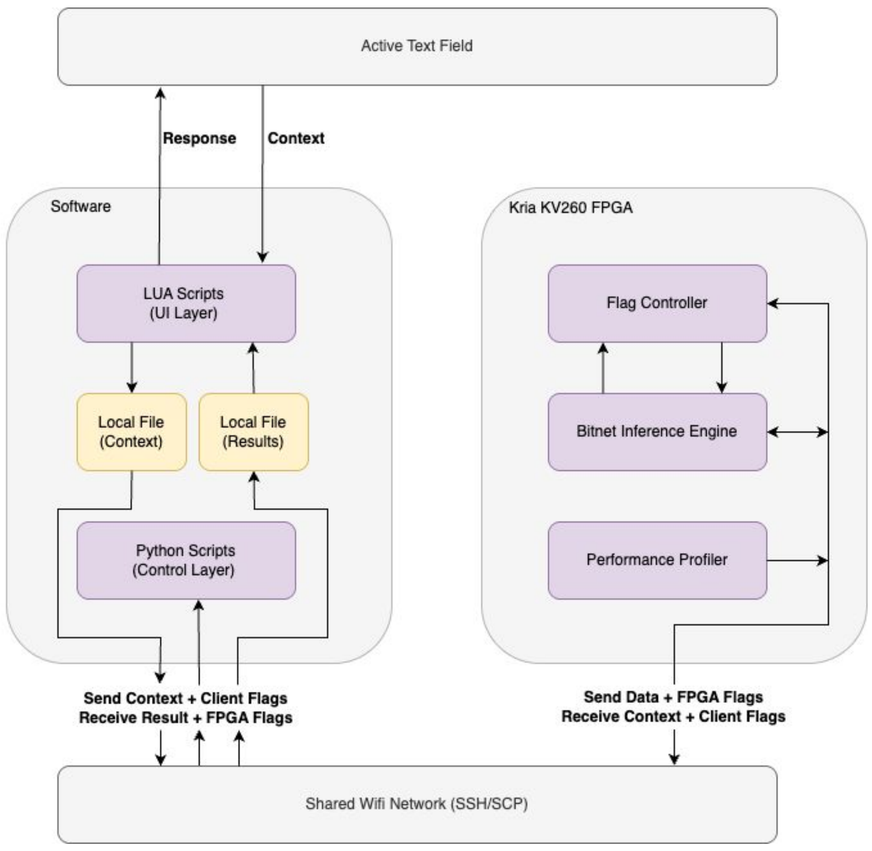
Product Pitch

AI co-pilots for text and code completion have become widespread, but most send user data to remote servers for processing, raising significant security concerns. As a result, many users — such as developers and journalists — are unable to use these tools. Their only alternative is to run models locally, but standard laptop hardware is poorly suited for machine learning inference, leading to slow performance and high energy consumption.

To address this gap, we developed an **FPGA-based** hardware accelerator capable of running 1.58-bit LLM variants known as **BitNets**. We also built a user interface that uses macOS scripting tools to trigger text or code suggestions in any text box via **customizable hotkeys**. Our on-device accelerator provides a competitive, privacy preserving alternative to current text completion services.

System Architecture

Our high-level system architecture consists of the Kria KV260 FPGA on the hardware side and client-side scripts written in Python and Lua. The Lua script captures user input from any active text field when a hotkey is pressed. The Python script establishes an SSH connection with the FPGA to monitor device readiness and uses secure copy (SCP) over a shared WiFi network to transfer input data and retrieve inference results. The FPGA stores our BitNet model and performs inference using a lookup-table-based arithmetic approach optimized for resource efficiency. In parallel, a performance profiler reads power consumption metrics from the PMBus monitors, returning these measurements alongside the inference output.



Conclusions & Additional Information

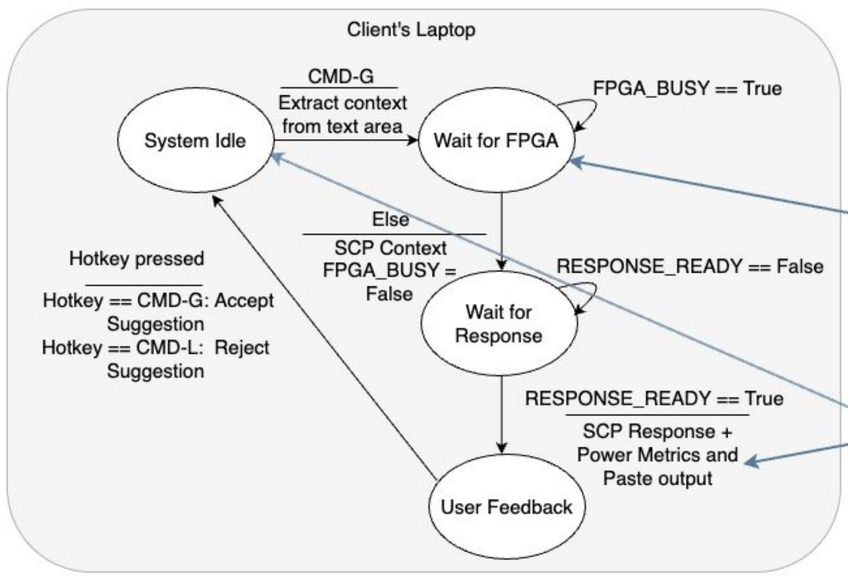
Our implementation successfully demonstrates the viability of FPGA-accelerated text completion on a personal device. Although the power savings over local CPU inference were modest, the significantly higher tokens-per-second throughput justifies the use of an off-board FPGA, similar to how data centers prioritize throughput over strict energy efficiency. Our system achieved notable improvements over local baselines, where typical Mac performance was limited to 4–5 tokens per second.

Future work includes enabling a headless Python client, supporting autocomplete at arbitrary cursor locations, and allowing customizable context window sizes to further improve flexibility and user experience.

System Description

Hardware: The Kria board runs the BitNet b1.58 LLM using quantized 1.58-bit inference kernels optimized for energy-efficient computation. Inference is performed via a lookup-table (LUT)-based arithmetic approach, where preloaded LUTs replace costly matrix multiplications. TL1/TL2 kernels further reduce bandwidth by compressing full-precision weights.

Software:



An SSH connection is established with the FPGA via a Python script for two purposes:

To authenticate the user, ensuring they are connected to the CMU WiFi network;

To read the FPGA busy flag from an internal register, indicating when it is ready to receive context.

Hammerspoon-based automation (Lua) enables our program to idle in the background until a hotkey is pressed.

Upon activation, it captures context from the active text field and transmits it over a TCP connection to the Python client scripts.

In Use:

```
test.py > ...
1  print("Starting loop...")
2
3  for i in range(1, 6):
4      if i % 2 == 0:
5          print(f"{i} is even")
6      else:
7          print(f" <<<PREVIEW>>> i is odd<<<END>>>")
```

System Evaluation

Use Case Metrics

Requirement	Success Criteria	Outcome	Met?
Latency less than CPU on a Mac	Tokens / second > 8	Tokens/sec: 15	Yes
Throughout less than CPU on a Mac	Time to first token < 250ms	Time to first token: 250ms	Yes
Power consumption less than CPU on a Mac	Power draw < 700 mW	Power: ~670mW	Yes
Simple installation process	Time to download < 15 mins	Time to download: < 2 mins	Yes

Tradeoff Analysis

Area	Decision	Impact
Latency	Moderate refresh rate to limit CPU usage.	Slightly higher response time, low CPU load
Starvation	First-come, first-serve without full queue	Occasional delays simpler client side logic
Model Size	Use a small BitNet b1.58 Model over quantized DeepSeek	Real time speeds, slightly lower output quality

Electrical & Computer ENGINEERING
Carnegie Mellon