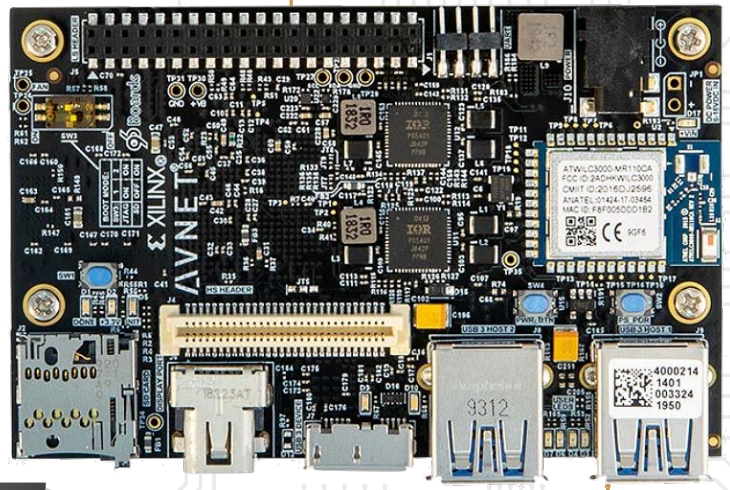
Texcelerate

Andrew Liao, Amelia Heller, Anirudh Prakash



Use Case + Requirements

Problem: Sending data to the cloud can be risky

- There is a range of users who cannot use commercial AI copilots because they work with sensitive data

Solution: On-device text and code completion

- FPGA accelerator for faster and more power efficient text & code completion
- UI that allows user to generate text on any text box on their Mac

Requirements:

Throughput	User Interface	General
To achieve instantaneous generation, tokens per second > 10	User should be able to choose whether or not to autocomplete text Setup/Installation should take the user less than 15 minutes	The system should support up to three wireless clients simultaneously

Technical Design Requirements

Quantified Requirement	Justification
Power consumption < 700 mW	600 - 700 mW on the CPU and 24-40 mW on the GPU
Time to first token < 250 ms **Tokens / second > 8**	Mean timing for text generation for our model run on a Mac is 1.11 - 1.3 seconds. This is less than what the human eye perceives as instant.
Context Window > 100 tokens	Anything less than this will truncate details, leading to less coherent completions

Design Tradeoffs

FPGA

Picked the Ultra96v2 over the Kria KV260 because the tool flow for the Ultra is easier to use

Also chose it over the ZedBoard because the ultra has more RAM and a hardcore

Platform

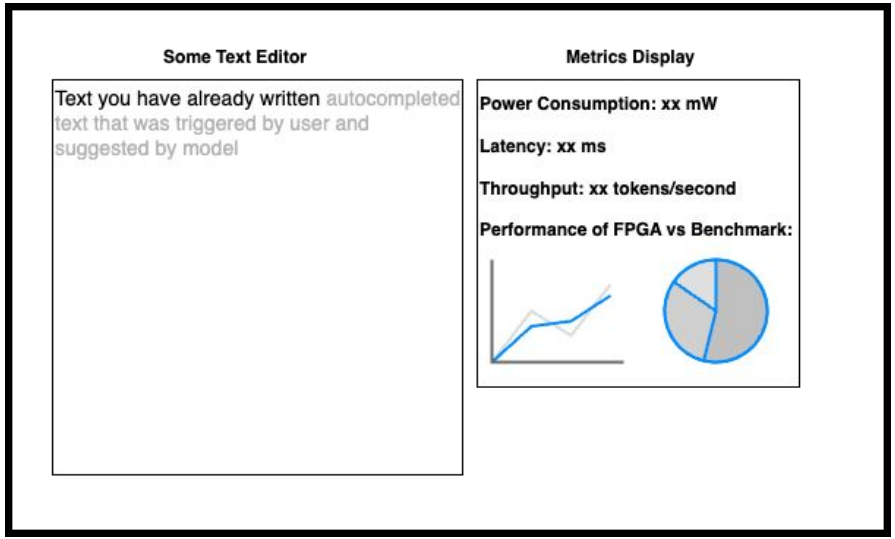
Our UI only works on Macbooks because it takes advantage of a MacOS specific tool to utilize keyboard interrupts.

Not accessible for all users but mitigates risk of low quality UI - since we are not software people.

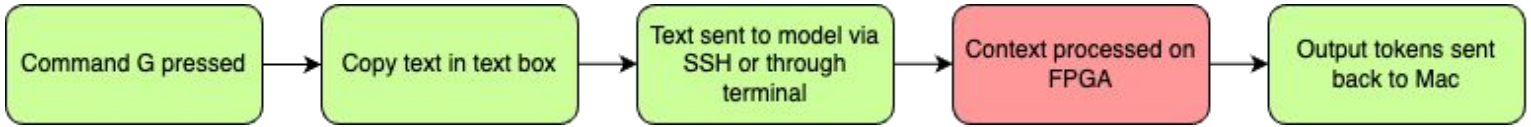
Model

Considered a quantized DeepSeek model over our current model but it's too difficult to prompt it for text completion and chain of thought output doesn't help fulfill use case requirements.

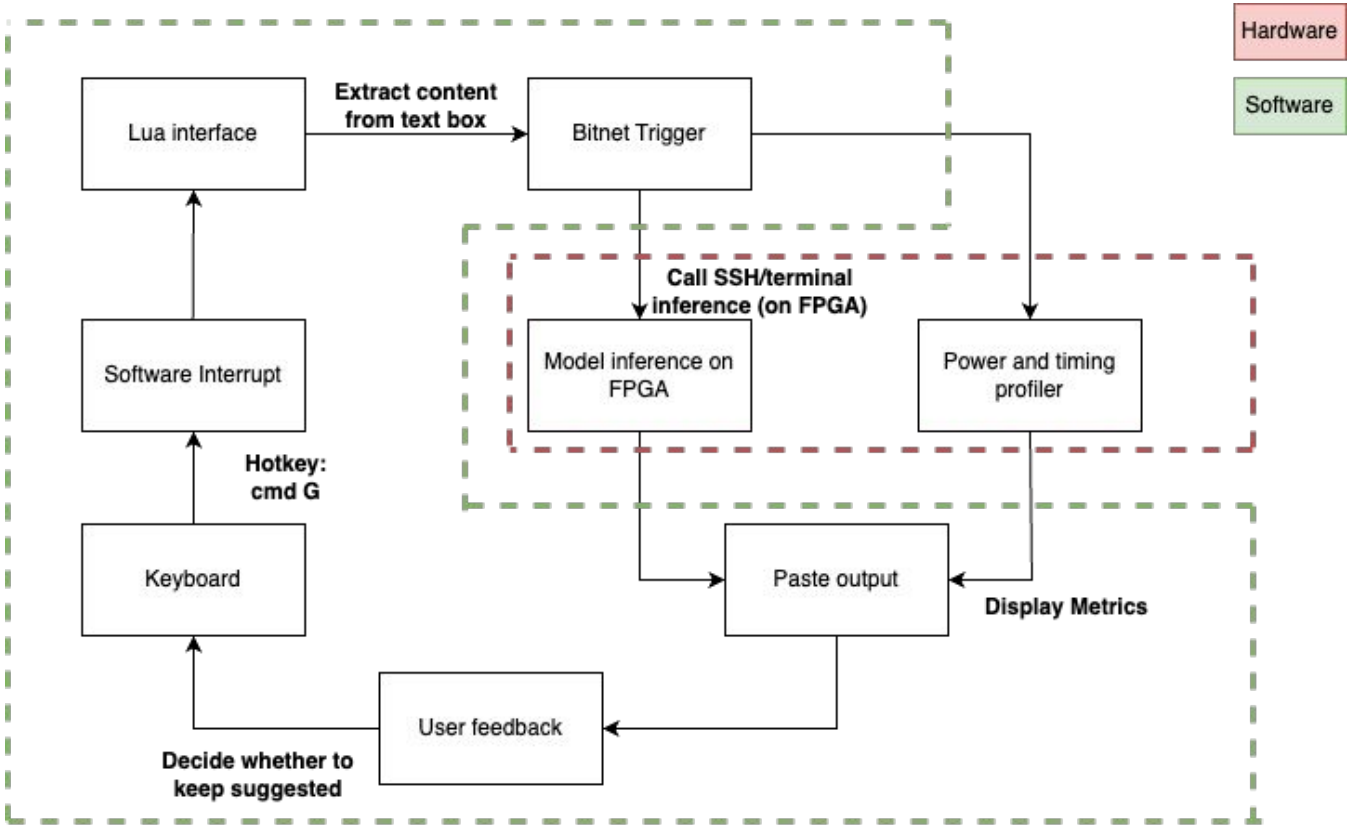
Solution Approach



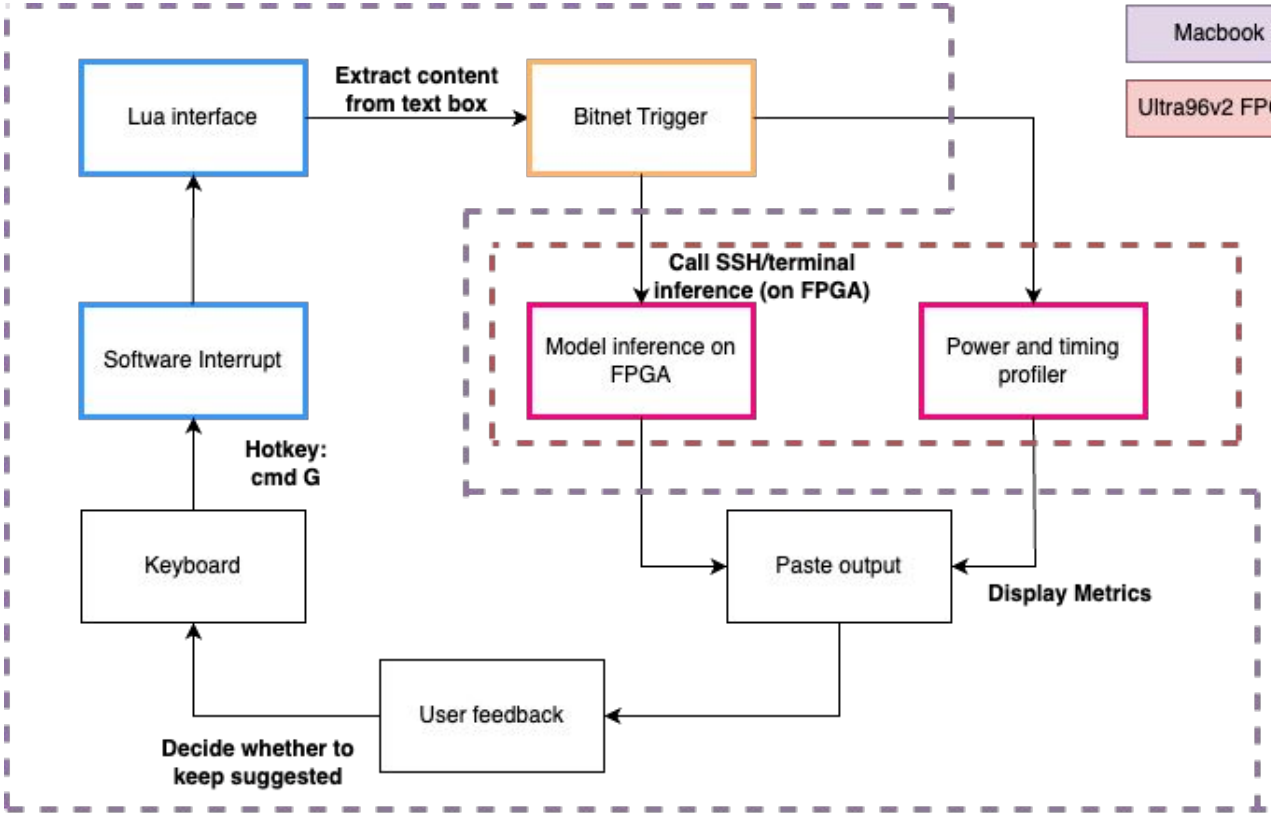
System Workflow:



System Specification - Block Diagram



Implementation Plan



Macbook

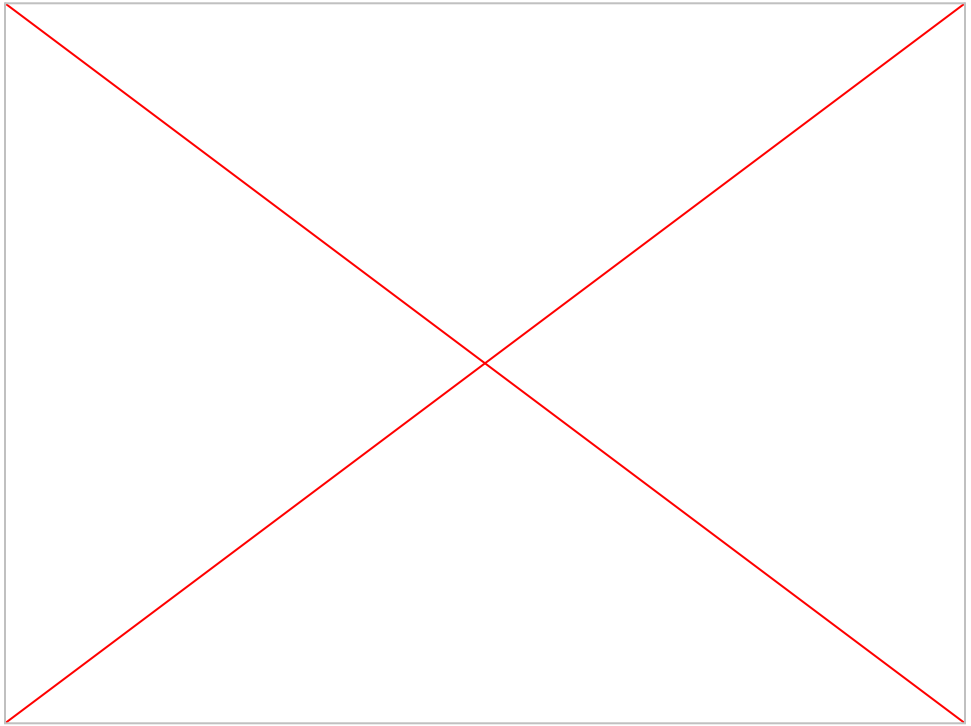
Ultra96v2 FPGA

Python to control hotkey and send context to model coded by us

Trained Bitnet model from Huggingface

RTL designed and implemented by us

UI Demo



Testing: Verification & Validation

Requirement	Test Method	Success Criteria
User can choose whether or not to accept generated text	Users – CMU Community Metrics – 1) How much time it takes users to download our system 2) Frequency with which the system let them override or accept text completions.	Text completions are overridden as requested 100% of the time
User can download and install system in < 15 minutes		90% of users taken < 15 minutes to install system The 10% who need more time should still be able to set up the system in < 25 minutes - This accounts for people who struggle to use technology
At least three users can connect to the accelerator wirelessly	We will attempt to all connect to the FPGA and run queries to it simultaneously.	Output quality of model is consistent across user and power and timing requirements hold

Testing: Verification & Validation

Requirement	Test Method	Success Criteria
Latency and throughput less than CPU & GPU on a Mac	On Mac – Power and Timing Profiler On FPGA – counters synthesized onto the fabric of the FPGA.	Tokens / second > 8 Time to first token < 250 ms
Power consumption less than CPU & GPU on a Mac	We will measure power consumption by interfacing with the PMIC on the Ultra96v2 FPGA	Power consumption < 700 mW

Testing: Risks & Ethical Concerns

Challenge	Mitigation Strategy
Broken wifi on our FPGA	We will try a wired UART connection to get around this. Worst case scenario is switch to the Kria KV260 FPGA
Limited FPGA iteration speed	We need to develop a synthesis flow. This is complicated if we switch to the KV260 which requires us to use Vitis
Multi-user security concerns	We need to justify allowing multiple users to use the same hardware to run text completion on sensitive information
Hallucinations & Biased Outputs	Our model scored a 30 on the truthfulQA benchmark and a 35.1 on the HellaSwag benchmark

Schedule

Texccelerate

Anirudh Prakash, Amelia Heller, Andrew Liao

Project start: **Wed, 1/29/2025**

Display week: **1**

TASK	ASSIGNED TO	PROGRESS	START	END
Project Planning				
Decide on FPGA		80%	1/29/25	2/5/25
Decide UI/UX Structure		100%	2/5/25	2/7/25
Decide Benchmark Softcores		100%	1/29/25	2/12/25
Choose Target Model		100%	1/29/25	2/3/25
ML Model				
Test existing BitNet Model		100%	1/29/25	2/1/25
Select Text model for quantize		100%	1/31/25	2/5/25
Quantize text model		100%	2/5/25	2/19/25
Modify inference code for CPU soft core deployment		100%	2/20/25	2/27/25
Modify inference code for GPU soft core deployment		0%	2/28/25	3/14/25
Modify inference code for FPGA deployment		0%	3/15/25	3/29/25
FPGA Acceleration				
Implement Unified Performance Counter		0%	2/19/25	2/26/25
Synthesize CPU/GPU soft cores		0%	2/19/25	2/26/25
Decide FPGA Architecture + RTL		0%	2/23/25	3/9/25
Verify Texccelerate RTL		0%	3/10/25	3/17/25
Synthesize Texccelerate on FPGA		0%	3/18/25	4/1/25
Synthesize Texccelerate on FPGA		0%	4/2/25	4/16/25
FPGA Interface, UI/UX				
Boot Linux on FPGA hard core		0%	2/28/25	3/3/25
FPGA to computer UART framework		0%	3/3/25	3/13/25
FPGA PS to PL communication framework		0%	3/14/25	3/24/25
Stream PMU metris through UART		0%	3/25/25	4/1/25
UI/UX Software Interface		100%	2/10/25	2/24/25
Whole System Integrated testing		0%	4/2/25	4/16/25