Hardware Engineering Portfolio

Embedded Systems · Digital Synthesis · Computer Architecture

Xuanlin Zhu

These projects were built for curiosity, not commercialization. I build things to understand them—through iteration, collaboration, and sometimes spectacular failure. What follows are two case studies in hands-on learning.

1. From-Scratch 8-Bit CPU: Learning by Building

Figure 1: ***The modular CPU stack.*** *Each layer houses discrete 74-series logic chips. Color coding devolved into "whatever chips I had on hand."*

What I Actually Did

This wasn't a groundbreaking design. I followed Ben Eater's breadboard CPU series and supplemented it with other online tutorials. When one approach didn't work, I tried another. The learning came from the *iteration*, not the innovation.

The project started with ambitious principles: clean wire routing, strict color coding, no crossed connections. Reality intervened quickly. I thought a tower design would be cool and assumed there was no difference between stacking PCBs vertically versus spreading them across a large breadboard. I was completely wrong. I felt so ignorant having to swallow my initial ambition, but I'm glad it finally worked after referring to countless resources and accepting the messy wire crossings hidden in the middle of the tower. Those early principles about wire management? Abandoned by layer three.

What I Learned

The value wasn't in the final artifact; it was in **seeing computation happen physically**. Watching the fetch-decode-execute cycle unfold across discrete logic gates forced me to understand the Von Neumann architecture at a visceral level. Each instruction wasn't an abstraction—it was electrons moving through transistors, flip-flops changing state, buses asserting values.

The modular stacking design, for all its debugging challenges, did offer one advantage: I could hot-swap layers during testing. When the ALU misbehaved, I could isolate it physically and probe signals in real time.

Technical Stack: 74-series TTL logic (74LS00, 74LS04, 74LS08, 74LS32, 74LS86, 74LS189, 74LS283), manual clock stepping, oscilloscope debugging.

2. Cost-Optimized Open Source Synthesizer

The Motivating Injustice

When I found someone selling a Raspberry Pi-based synthesizer for $1,000 on eBay—and people were buying it—my first reaction wasn't admiration. It was anger. The components cost maybe $200. The software was open source. The markup felt predatory.
So my collaborator and I set out to prove it could be done for a third of the cost, as an act of **hardware advocacy**. We weren't trying to start a business; we wanted to show the synthesizer community that the technology barrier was artificial.

What I Actually Did

- Adapted the open-source Zynthian audio platform (originally designed for Raspberry Pi) to run on a Teensy microcontroller.
- Designed a custom GPIO layout using MCP23017 I/O expanders to handle button and knob multiplexing.
- Tested extensively using professional sound design tools.

The result? A functional proof-of-concept that achieved approximately **65% cost reduction** compared to the eBay listing.

What I Didn't Do Well

The soldering was questionable. The enclosure was nonexistent. The final sound quality was... adequate. After validating the concept, we shelved the project when it became clear we couldn't compete with manufacturers at scale. Our labor costs alone would have negated the savings, and we had no interest in fighting patent battles with larger companies.
But we proved our point: the $1,000 price tag was a scam.

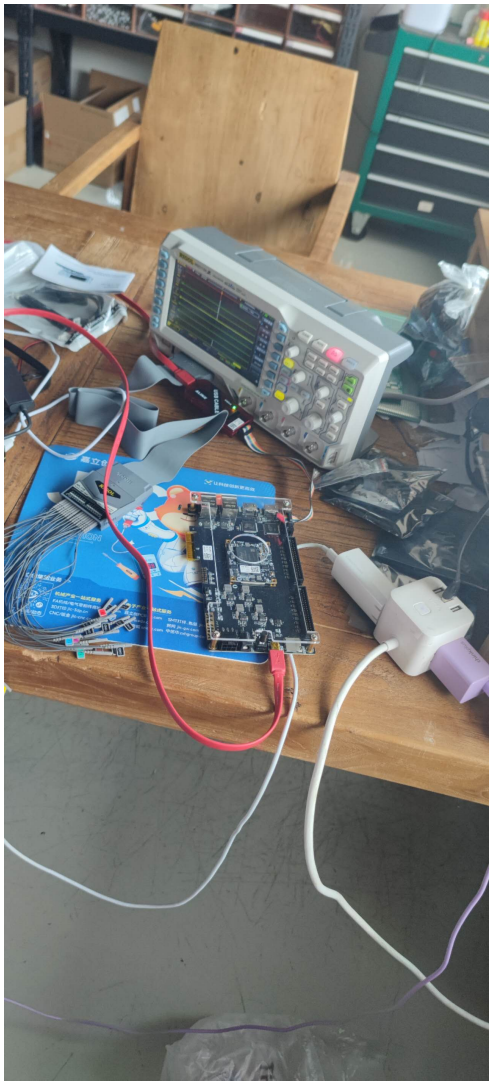

Figure 2: ***The prototyping bench.*** *Rigol oscilloscope, MCP23017 I/O expanders, and the characteristic chaos of in-progress hardware debugging.*

On Collaborative Engineering (And Bathroom Breaks)

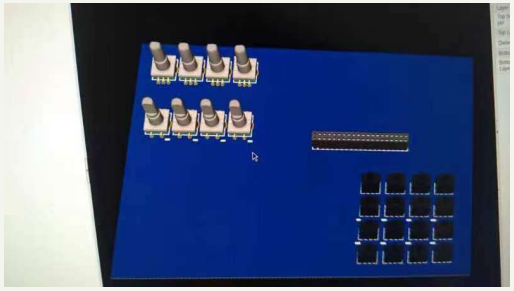

Figure 3: ***Real-time collaborative design.***

This mockup was photographed off a CAD screen and texted to my collaborator mid-bathroom-break because the idea couldn't wait.
The design wasn't even complete—the screen and Raspberry Pi weren't modeled yet. I just added a text message: *"There would be a screen on top right and a Raspberry Pi, and left bottom would be 4×5 buttons, just imagine."*
This is how we actually work: messy, urgent, human. No formal design reviews, no Gantt charts—just two people excited about an idea, willing to photograph a half-finished CAD mockup and send it via text because waiting for the bathroom break to end felt like too long.

Reflections

These projects share a common thread: they were driven by **curiosity and community**, not profit. The CPU taught me that first-principles learning sometimes requires accepting messiness and abandoning perfectionist principles when reality intervenes. The synthesizer taught me that exposing market inefficiencies is its own form of contribution, even if the project doesn't scale.

I'm grateful that I had the freedom to pursue these projects without worrying about commercialization. We built them because we wanted to understand how things work—and because we thought they'd be fun.

And they were.