It’s exciting that you’re diving deeper into this analog computing concept, especially with the idea of energy-signature-based logic gates. While it's natural to have hesitations about whether someone else has explored this or if there’s an inherent flaw, I think it’s important to approach it with the mindset that even if parts of the idea exist, your implementation could still be novel. And sometimes, the best breakthroughs come from building on existing ideas in unexpected ways.

Let’s break this down:

Energy-Signature-Based Logic: This concept where power (energy signatures) themselves carry both the instruction and the “data” is fascinating. I think your hesitation about signal interference is valid. In traditional electronics, signal degradation or interference happens frequently due to noise, especially as you scale up a system. However, it’s possible that, by carefully designing how energy is modulated and how signals are interpreted, you could avoid some of these pitfalls. You might want to look into signal processing techniques used in radio transmission or fiber optics—industries that deal with modulated signals over long distances with minimal loss.

Emulating Binary Logic: Emulating a binary system using an analog framework is a clever way to bootstrap into familiar territory. You’re essentially setting up the system to “speak the same language” as traditional digital systems, which could be key in making your analog setup more interoperable with existing tech. However, the real power of analog computation lies in continuous variables, not just on/off states. If you can solve the challenge of implementing basic binary logic gates, you could later explore new functionalities that take advantage of the analog nature, such as handling real numbers, signal filtering, or even solving differential equations natively.

Memory and Machine-Level Code: You're absolutely on track with bootstrapping into memory and machine code. In digital systems, memory is often binary-based (1s and 0s stored in cells), but in an analog system, memory could potentially take the form of charge, voltage, or frequency variations. Figuring out how to store and retrieve data in a robust, noise-resistant way could be a big challenge, but once solved, it opens the door to analog programs, algorithms, or even hybrid approaches where data is processed in analog but stored in a digital or semi-digital format.

Signal Interference and Noise: This is a known challenge in analog systems, and you’re right to focus on it. Potential solutions might include:

Error correction: Borrow techniques from communications, like error-correcting codes, to ensure that slight variations in energy signals don’t corrupt your data.

Shielding and Isolation: At a hardware level, ensuring that signals don’t interfere with each other by isolating different parts of the circuit might be crucial.

Signal Amplification: Similar to how op-amps are used in analog computers, amplifying the correct signal while filtering out noise could be necessary.

Analog Programs: The idea of running an analog "program" or "algorithm" is very intriguing. Traditional analog computers were often used for simulations (e.g., solving differential equations by physically modeling systems), but you’re suggesting something more dynamic. If you can find a way to dynamically change configurations (without needing manual rewiring), you open up the possibility for a programmable analog system. Modern technology might help

here—MEMS (micro-electro-mechanical systems) or nanoscale switches could be part of a solution where the system reconfigures itself on the fly based on the incoming signals.

A Few Guiding Questions:

How would you define the energy signatures in a way that can reliably carry out computation? Would frequency, voltage, or phase shifts in the signal represent different instructions?

Can the system differentiate between data and instructions, and how would it "know" which energy signature is carrying which?

What kind of feedback loop would the system need to self-correct, especially with interference or noise?

Potential Next Steps:

Simulations: Before building physical prototypes, simulating the system might help test its feasibility and allow you to fine-tune signal processing. Tools like LTSpice, or even MATLAB/Simulink, could give you insights.

Prototyping Simple Gates: Start small by designing a basic NAND gate that works via modulated energy inputs. Once you have that, you could explore more complex gates and arithmetic functions.

Exploring Hybrid Analog-Digital Architecture: Since emulating binary logic seems natural, a hybrid approach could serve as a bridge between analog computations and existing digital systems. This would give you the flexibility to handle complex tasks (e.g., encryption, floating-point arithmetic) while still maintaining an analog foundation.

I think your analog computer can have a lot of potential beyond just emulating binary logic, especially if you push the envelope on where analog computation could outperform digital systems. I'm excited to see how you proceed with this—especially when you start experimenting with the practical side of things. Keep challenging these assumptions and pushing through the uncertainty!