Analog system architecture

Exploring the system architecture for analog computing is exciting, and since it's relatively uncharted territory, we have the flexibility to rethink traditional structures.

1. Foundation of Analog Logic

Analog Signals as Data: Instead of binary 0s and 1s, analog signals represent continuous values, such as varying voltages, currents, or frequencies.
Thresholding: In binary logic, a gate outputs a 1 or 0 based on specific input thresholds. In an analog system, gates might output varying signal levels based on input signal strengths. The challenge is in defining and interpreting these thresholds or creating gates that modulate signal strength as needed.

Example:

In a NAND gate, two inputs would normally produce a binary result (0 or 1). In an analog version, we'd instead interpret a range of voltages as a "not and" response. So, a low voltage might be seen as a 0, and a higher voltage as a 1.

2. Gate Design in Analog Computing

Transistors: In analog computing, transistors still form the core of most gates, but the way they respond to signal strengths differs. You'll likely use a mix of transistors, resistors, capacitors, and op-amps to manage voltage ranges and perform thresholding for analog logic.
Operational Amplifiers (Op-Amps): Op-Amps are useful in analog systems because they can amplify weak signals or compare two voltage levels. They can be used to create analog versions of digital logic gates.

For instance, a basic analog **NAND gate**:

Two input voltages are combined, and the Op-Amp determines the output based on whether the combined voltage exceeds a certain threshold. You can also adjust resistances or capacitances to fine-tune these thresholds.

3. Memory and Storage in Analog Systems

In traditional computing, **flip-flops** and **latches** are used to store binary values. For analog systems, memory could be stored as charge on capacitors or in feedback loops using Op-Amps.
Capacitors: Capacitors can hold charge at specific levels, which might represent the value stored in the system. Instead of a binary flip-flop, you could have a range of charge levels representing a range of values.
Feedback Loops: Using feedback in an Op-Amp circuit, you could store an analog signal as part of the system's state. This would allow the system to remember values without relying on binary data storage.

4. Processing and Arithmetic

Analog Arithmetic: Unlike binary-based arithmetic, which relies on discrete states, analog arithmetic works by directly manipulating signal strengths. For example, you could sum signals by combining voltages, multiply them by varying resistance, or subtract them using differential circuits.
Analog ALU (Arithmetic Logic Unit): An analog ALU would function by manipulating signal voltages. Operations like addition, subtraction, and integration could be performed by combining or differentiating the analog signals.

5. Power as a Data Stream

Modulated Power as Input: As you mentioned before, power could carry data. The system would need components designed to interpret modulated signals (voltage or frequency modulation) as instructions.

Gate Activation and Control: The system would need circuits designed to interpret the embedded data in the power stream. These would act as decoders, interpreting signals and executing corresponding operations (activating logic gates or running sequences).

For example, a modulated power signal could be sent to the gates, and the gates interpret the signal as both a power source and an operational instruction.

6. Self-Configuring System

Probing the System: One core idea you mentioned was sending a signal to a device and probing for available components.
When probing, the system could send a series of modulated signals to each potential connection point. Each component would have a "response signature" that tells the system what it is and how to interact with it.
The system could then activate its core processor (analog ALU) to begin running operations that take advantage of each detected component.

7. Practical Considerations

Low Power Testing: As you pointed out, analog systems tend to use less power. This opens up the possibility of running these tests with simple power sources (like AA batteries) before scaling to more complex systems with lab-grade supplies.
Modular Design: Each of the circuits we design can be modular. We can focus on individual gates and components first, ensuring they work, and then start connecting them to form larger systems, such as the processor, memory, and I/O subsystems.

Immediate Next Steps:

Designing and Prototyping Gates:

Continue testing basic gates (NAND, AND, OR, etc.) using analog components like Op-Amps and transistors.
Simulate gate behavior using both theoretical models and practical components (e.g., resistors to create threshold effects).

Basic Arithmetic and Memory:

Use basic analog components (capacitors, transistors) to design simple memory systems and start experimenting with basic arithmetic units.

Self-Configuring Exploration:

Think about how the probing mechanism would work. What would the power/data signal look like? How would it interact with the components on the board?