An Analog Accelerator for Linear Algebra

Yipeng Huang, Ning Guo, Mingoo Seok, Yannis Tsividis, Simha Sethumadhavan

Columbia University
Why Analog?

Digital algorithms

- Binary numbers
- Step-by-step operation

Digital hardware

Supports
Why Analog?

Digital algorithms

Digital hardware

Supports

Scaling & architecture

Faster, more efficient digital hardware
Why Analog?

- Digital algorithms
  - Supports
  - Research & development

- Digital hardware
  - Supports
  - Scaling & architecture

- More optimal digital algorithms
  - Supports
  - Faster, more efficient digital hardware
Why Analog?

Digital hardware

Gates’ Simplification #1:
Dennard scaling ended, Moore scaling will end
—Monday keynote speaker Doug Carmean

Digital algorithms
Research & development

More optimal digital algorithms
_scaling & architecture

Supports

Faster, more efficient digital hardware
Why Analog?

Digital algorithms

Research & development

Scaling & architecture

More optimal digital algorithms

Supports

Faster, more efficient digital hardware

Supports

Digital hardware

Analog hardware
Why Analog?

Digital algorithms → Research & development → More optimal digital algorithms

Digital hardware → Scaling & architecture → Faster, more efficient digital hardware

Analog algorithms

Analog hardware
Why Analog?

Digital algorithms

- Supports

Digital hardware

Research & development

- Supports

More optimal digital algorithms

Scaling & architecture

Faster, more efficient digital hardware

Analog algorithms

- No binary numbers
- Continuous operation

Analog hardware

- Supports
A continuous-time, analog computing model
• step-by-step algorithm $\rightarrow$ continuous-time algorithm
• continuous-time algorithm $\rightarrow$ analog accelerator hardware

Analog drawbacks: how to fix them

A prototype analog accelerator & evaluation
Analog computing solves ordinary differential equations
Scientific computation phrased problems as ODEs

Modern problems are converted to linear algebra, not ODEs
Can we accelerate linear algebra using analog?
Ax = b
\[ Ax = b \]

\[
\begin{align*}
    a_{00}x_0 + a_{01}x_1 &= b_0 \\
    a_{10}x_0 + a_{11}x_1 &= b_1
\end{align*}
\]
\[ Ax = b \]

\[
\begin{align*}
    a_{00}x_0 + a_{01}x_1 &= b_0 \\
    a_{10}x_0 + a_{11}x_1 &= b_1
\end{align*}
\]

Direct methods
- E.g., Gaussian elimination

Iterative methods
\[ Ax = b \]

\[
\begin{align*}
  a_{00}x_0 + a_{01}x_1 &= b_0 \\
  a_{10}x_0 + a_{11}x_1 &= b_1
\end{align*}
\]

**Direct methods**
- E.g., Gaussian elimination

**Iterative methods**
- E.g., steepest gradient descent
- E.g., conjugate gradients
Solve \begin{align*}
    a_{00}x_0 + a_{01}x_1 &= b_0 \\
    a_{10}x_0 + a_{11}x_1 &= b_1
\end{align*}

Step-by-step

steepest gradient descent
Solve \[
\begin{aligned}
\begin{cases}
a_{00}x_0 + a_{01}x_1 = b_0 \\
a_{10}x_0 + a_{11}x_1 = b_1
\end{cases}
\end{aligned}
\]

**Step-by-step**

steepest gradient descent

recurrence relation

\[
\begin{aligned}
\begin{cases}
x_0^{n+1} = x_0^n - s(a_{00}x_0^n + a_{01}x_1^n - b_0) \\
x_1^{n+1} = x_1^n - s(a_{10}x_0^n + a_{11}x_1^n - b_1)
\end{cases}
\end{aligned}
\]
Solve \[
\begin{aligned}
a_{00}x_0 + a_{01}x_1 &= b_0 \\
a_{10}x_0 + a_{11}x_1 &= b_1
\end{aligned}
\]

**Step-by-step**

steepest gradient descent

recurrence relation

\[
\begin{aligned}
x_0^{n+1} &= x_0^n - s(a_{00}x_0^n + a_{01}x_1^n - b_0) \\
x_1^{n+1} &= x_1^n - s(a_{10}x_0^n + a_{11}x_1^n - b_1)
\end{aligned}
\]

**Continuous-time**

continuous steepest descent
Solve \[ \begin{align*} a_{00}x_0 + a_{01}x_1 &= b_0 \\ a_{10}x_0 + a_{11}x_1 &= b_1 \end{align*} \]

**Step-by-step**

steepest gradient descent

recurrence relation

\[
\begin{align*}
x_0^{n+1} &= x_0^n - s(a_{00}x_0^n + a_{01}x_1^n - b_0) \\
x_1^{n+1} &= x_1^n - s(a_{10}x_0^n + a_{11}x_1^n - b_1)
\end{align*}
\]

**Continuous-time**

continuous steepest descent

ordinary differential equation

\[
\begin{align*}
dx_0/dt &= -a_{00}x_0 - a_{01}x_1 + b_0 \\
dx_1/dt &= -a_{10}x_0 - a_{11}x_1 + b_1
\end{align*}
\]
CONTINUOUS-TIME

Potentially fast: not limited by step-by-step algorithm
A continuous-time, analog computing model
  • step-by-step algorithm $\rightarrow$ continuous-time algorithm
  • continuous-time algorithm $\rightarrow$ analog accelerator hardware

Analog drawbacks: how to fix them

A prototype analog accelerator & evaluation
Analog accelerator hardware

Datapath: explicit data flow

Values: represented as analog current & voltage

Functional units: analog arithmetic operators
Solve \[
\begin{align*}
  a_{00}x_0 + a_{01}x_1 &= b_0 \\
  a_{10}x_0 + a_{11}x_1 &= b_1
\end{align*}
\]

ordinary differential equation
\[
\begin{align*}
  \frac{dx_0}{dt} &= -a_{00}x_0 - a_{01}x_1 + b_0 \\
  \frac{dx_1}{dt} &= -a_{10}x_0 - a_{11}x_1 + b_1
\end{align*}
\]

Integrators
\[ \begin{align*}
\int dx_0 &= x_0(t) - a_{00} \\
\int dx_1 &= x_1(t) - a_{11}
\end{align*} \]

ordinary differential equation
\[ \begin{align*}
\frac{dx_0}{dt} &= -a_{00}x_0 - a_{01}x_1 + b_0 \\
\frac{dx_1}{dt} &= -a_{10}x_0 - a_{11}x_1 + b_1
\end{align*} \]
ordinary differential equation
\[
\begin{align*}
\frac{dx_0}{dt} &= -a_{00}x_0 - a_{01}x_1 + b_0 \\
\frac{dx_1}{dt} &= -a_{10}x_0 - a_{11}x_1 + b_1
\end{align*}
\]
ordinary differential equation
\[
\begin{align*}
\frac{dx_0}{dt} &= -a_{00}x_0 - a_{01}x_1 + b_0 \\
\frac{dx_1}{dt} &= -a_{10}x_0 - a_{11}x_1 + b_1
\end{align*}
\]
ordinary differential equation
\[
\begin{align*}
\frac{dx_0}{dt} &= -a_{00}x_0 - a_{01}x_1 + b_0 \\
\frac{dx_1}{dt} &= -a_{10}x_0 - a_{11}x_1 + b_1
\end{align*}
\]

Sum currents by joining wires
ordinary differential equation

\[
\begin{align*}
\frac{dx_0}{dt} &= -a_{00}x_0 - a_{01}x_1 + b_0 \\
\frac{dx_1}{dt} &= -a_{10}x_0 - a_{11}x_1 + b_1
\end{align*}
\]
ordinary differential equation
\[
\begin{align*}
\frac{dx_0}{dt} &= -a_{00}x_0 - a_{01}x_1 + b_0 \\
\frac{dx_1}{dt} &= -a_{10}x_0 - a_{11}x_1 + b_1
\end{align*}
\]
Solve \[ \begin{cases} a_{00}x_0 + a_{01}x_1 = b_0 \\ a_{10}x_0 + a_{11}x_1 = b_1 \end{cases} \]

This is an ordinary differential equation:
\[ \begin{cases} \frac{dx_0}{dt} = -a_{00}x_0 - a_{01}x_1 + b_0 \\ \frac{dx_1}{dt} = -a_{10}x_0 - a_{11}x_1 + b_1 \end{cases} \]
CONTINUOUS-TIME
Potentially fast: not limited by step-by-step algorithm

ANALOG VALUES
Potentially efficient: one wire carries real number
A continuous-time, analog computing model

Analog drawbacks:
- limited applications:
- limited accuracy:
- limited precision:
- limited scalability:

how to fix them
- tackle key linear algebra kernel calibration & exceptions
- build precision with digital help divide & conquer sparse matrix

A prototype analog accelerator & evaluation
Accuracy

Digital
Intermediate values unambiguously interpreted as 1 or 0

Analog
Process & temperature variation → computation result variation!
### Accuracy

**Digital**
- Intermediate values unambiguously interpreted as 1 or 0
- Discrete math error correction possible

**Analog**
- Process & temperature variation $\rightarrow$ computation result variation!
- Within purely analog execution, no error correction
Accuracy: calibration & exceptions

Components are inaccurate:

- Gain
Components are inaccurate:

- Gain
- Offset
Accuracy: calibration & exceptions

Components are inaccurate:

• Gain
• Offset
• Nonlinearity (clipping)
Components are inaccurate:

- Gain
- Offset
- Nonlinearity (clipping)

Calibrate: all components using additional DACs
Components are inaccurate:
- Gain
- Offset
- Nonlinearity (clipping)

Calibrate: all components using additional DACs

Exceptions: catch values exceeding valid input range
A continuous-time, analog computing model

Analog drawbacks:
- limited applications:
- limited accuracy:
- limited precision:
- limited scalability:

how to fix them
- tackle key linear algebra kernel calibration & exceptions
- build precision with digital help divide & conquer sparse matrix

A prototype analog accelerator & evaluation
Precision: build precision w/ digital help

Limitation in sampling resolution
→ limited precision result
Precision: build precision w/ digital help

Limitation in sampling resolution
→ limited precision result

Find residual, rescale problem, & solve again in analog for precision
A continuous-time, analog computing model

Analog drawbacks:
- limited applications:
- limited accuracy:
- limited precision:
- limited scalability:

how to fix them
- tackle key linear algebra kernel calibration & exceptions
- build precision with digital help divide & conquer sparse matrix

A prototype analog accelerator & evaluation
Without time multiplexing, analog hardware needed for all variables

- Impractical to build analog hardware for whole problems

Scalability: divide & conquer sparse matrix
Without time multiplexing, analog hardware needed for all variables

- Impractical to build analog hardware for whole problems

Solve subproblems of smaller size in analog

- Possible because many problems have sparse matrices
A continuous-time, analog computing model

Analog drawbacks: how to fix them

A prototype analog accelerator & evaluation
- microarchitecture
- architecture & programming
- energy
- performance
Prototype analog accelerator: why & how

Why prototype?
- validate analog circuits
- physical measurement of components
- prototype helps developing analog applications

Engineering process
- 2 full-time PhD students, 4 project students
- 3 years
- full custom analog
- synthesized digital
- I worked on validation, digital interface, applications
Prototype analog accelerator: μArch.

Components
• 20 KHz analog bandwidth
• 4 integrators
• 8 multipliers
• Other features: nonlinear lookup

Fabrication
• 1.2 V 65nm TSMC
• Low power density
  • 0.06 W/cm² at full power
  • 2.0 mm² active area
  • 1.2 mW at full power
Prototype analog accelerator: interface

Architecture
- 8-bit A/D/A conversion
- Configurable analog crossbar
- Calibration & exceptions on all analog

Programming
- Library for configuration
- ODE syntax parser and compiler
A continuous-time, analog computing model

Analog drawbacks: how to fix them

A prototype analog accelerator & evaluation
- microarchitecture
- architecture & programming
- energy
- performance
Solution energy: analog HW vs. digital SW
Solution time: analog HW vs. digital SW

![Graph showing comparison of digital and analog solution times. The graph plots number of variables on the x-axis and time to solution in μs on the y-axis. The digital solution time is represented by a blue line, and the analog solution time is represented by a green line. The graph indicates that for a given number of variables, the analog solution time is significantly lower (10x less) than the digital solution time.]
Solution time: analog HW vs. digital SW

- Analog:
  - Explicit dataflow architecture

Digital solution time

Analog solution time

Time to solution (μs)

Number of variables
Solution time: analog HW vs. digital SW

Analog:
- Explicit dataflow architecture
- Continuous-time speed
- Analog efficiency

Digital solution time

Analog solution time
Solution time: analog HW vs. digital SW

- Analog:
  - Explicit dataflow architecture
  - Continuous-time speed
  - Analog efficiency
  - High analog area cost

**Graph:**
- X-axis: Number of variables
- Y-axis: Time to solution (μs)
- Blue line: Analog solution time
- Green line: Digital solution time

As the number of variables increases, the time to solution for analog HW increases significantly compared to digital SW.
Solution time: analog HW vs. digital SW

Digital:
- Optimal digital algorithms

Analog:
- Explicit dataflow architecture
- Continuous-time speed
- Analog efficiency
- High analog area cost

Time to solution (μs)

Number of variables
An Analog Accelerator for Linear Algebra

Continuous-time + analog offers alternative abstractions to digital

Tackled analog drawbacks: generality, accuracy, precision, scalability

Analog prototype: ISA & hardware; speed, area, energy analysis

Should we use analog to accelerate linear algebra?
Some gains, but digital algorithms are optimized!
Analog promises greater advantage in other problems: nonlinear?
<table>
<thead>
<tr>
<th>Continuous time</th>
<th>Digital</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Analog</strong></td>
<td><strong>Digital</strong></td>
</tr>
<tr>
<td><strong>Continuous time</strong></td>
<td><strong>Discrete time</strong></td>
</tr>
<tr>
<td>Ordinary differential equation</td>
<td>Recurrence relation</td>
</tr>
<tr>
<td>Advantage: fast</td>
<td>Advantage: allows complex algorithms</td>
</tr>
<tr>
<td>Advantage: low power b/c no clock</td>
<td>Advantage: allows time multiplexing</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Continuous value</th>
<th>Discrete value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Analog</strong></td>
<td><strong>Digital</strong></td>
</tr>
<tr>
<td>Current &amp; voltage</td>
<td>Integers &amp; floating point</td>
</tr>
<tr>
<td>Advantage: fast and efficient operations</td>
<td>Advantage: high dynamic range</td>
</tr>
<tr>
<td>Advantage: one wire carries real number</td>
<td>Advantage: high signal-to-noise ratio</td>
</tr>
</tbody>
</table>