The Demise and Resurrection of Analog Computing

Yipeng Huang
PhD Candidacy Exam
April 8, 2015
My Research

- Hybrid continuous-discrete computing
- Robotics workload characterization
<table>
<thead>
<tr>
<th>Outline &amp; Purpose</th>
<th>Digital</th>
<th>Analog</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Workloads</strong></td>
<td>[2], [3], [9], [15], [22], [28]</td>
<td>[26]</td>
</tr>
<tr>
<td><strong>Programming model</strong></td>
<td>[14], [17], [21], [25]</td>
<td>[7], [8]</td>
</tr>
<tr>
<td><strong>Datapath &amp; network</strong></td>
<td>[1], [5], [13], [16], [19]</td>
<td>[4], [8]</td>
</tr>
<tr>
<td><strong>Memory hierarchy</strong></td>
<td>[9], [16], [18], [19], [22], [28]</td>
<td>[4], [26]</td>
</tr>
<tr>
<td><strong>Core organization</strong></td>
<td>[6], [16], [17], [24], [28]</td>
<td>[4]</td>
</tr>
<tr>
<td><strong>Logic &amp; pipelines</strong></td>
<td>[10], [11], [12], [16], [20], [22], [23], [24]</td>
<td>[8], [23], [26], [27]</td>
</tr>
<tr>
<td><strong>Physics &amp; devices</strong></td>
<td>[6]</td>
<td>[26]</td>
</tr>
<tr>
<td>Subsection</td>
<td>Digital</td>
<td>Analog</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>---------</td>
<td>--------</td>
</tr>
<tr>
<td>Workloads</td>
<td>[2], [3], [4], [22], [28]</td>
<td>[26]</td>
</tr>
<tr>
<td>Programming model</td>
<td>[14], [17], [21], [25]</td>
<td>[7], [8]</td>
</tr>
<tr>
<td>Datapath &amp; network</td>
<td>[1], [5], [8], [16], [19]</td>
<td>[4], [8]</td>
</tr>
<tr>
<td>Memory hierarchy</td>
<td>[9], [16], [19], [22], [28]</td>
<td>[4], [26]</td>
</tr>
<tr>
<td>Core organization</td>
<td>[6], [16], [24], [28]</td>
<td>[4]</td>
</tr>
<tr>
<td>Logic &amp; pipelines</td>
<td>[10], [11], [16], [20], [22], [23], [24]</td>
<td>[8], [23], [26]</td>
</tr>
<tr>
<td>Physics &amp; devices</td>
<td>[6]</td>
<td></td>
</tr>
</tbody>
</table>

Expose bottlenecks in digital & provide ideas for analog
Digital: Workloads & Programs

- In cloud workloads, less locality & predictability
  - Instructions, working data set exceeds capacity of fast cache [9], [19]

- Structured Parallelism is Decreasing
  - Less structured parallelism: scientific computation, linear algebra
  - More amorphous parallelism: graph traversal, dynamic programming
  - Requires new metrics for parallelism [21]

Workloads present decreasing traits that digital computers exploit
• **Optimal network design depends on workload**
  • Complexity may help: Balfour et.al. find meshed, repeated, heterogeneous, torus network optimal [1].
  • Or, it may not: Lotfi-Kamran et al. find a basic crossbar suitable for their design [19].
  • In domain specific or application specific accelerators, datapath can be further refined [22].
OoO Superscalar vs. GPU dichotomy has broken down
To reduce static power, must shut circuits off or use DVFS
  - Challenge is conveying to microarchitecture the desired throughput, latency, accuracy

Throughput
  - Tao parallelism analysis assists disabling pipelines & whole cores [21]

Latency
  - Slack analysis may guide dynamic voltage frequency scaling [10]

Accuracy
  - Benign data approximation permits using lower power pipelines [23], [26], [27]

Current ISAs communicate little info to guide energy conservation
Must minimize pipeline power spent outside of function units

- Custom instructions [12] [24], Dyser, ASICs [11] [22], all are attempts to minimize fetch, decode, pipeline, control, register power

Stored program computers incur high overhead outside of ALUs
<table>
<thead>
<tr>
<th>Outline &amp; Purpose</th>
<th>Digital</th>
<th>Analog</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workloads</td>
<td>[2], [3], [9], [15], [22], [28]</td>
<td>[26]</td>
</tr>
<tr>
<td>Programming model</td>
<td>[14], [17], [21], [25]</td>
<td>[7], [8]</td>
</tr>
<tr>
<td>Datapath &amp; network</td>
<td>[1], [5], [6], [13], [16], [19]</td>
<td>[4], [8]</td>
</tr>
<tr>
<td>Memory hierarchy</td>
<td>[9], [16], [18], [19], [22], [28]</td>
<td>[4], [26]</td>
</tr>
<tr>
<td>Core organization</td>
<td>[6], [16], [17], [24], [28]</td>
<td>[4]</td>
</tr>
<tr>
<td>Logic &amp; pipelines</td>
<td>[10], [11], [12], [16], [20], [22], [23], [24]</td>
<td>[8], [23], [26], [27]</td>
</tr>
<tr>
<td>Physics &amp; devices</td>
<td>[6]</td>
<td>[26]</td>
</tr>
</tbody>
</table>

Exposé bottlenecks in digital & provide ideas for analog
Digital Bottlenecks Summary

- Workloads present decreasing traits that digital computers exploit
- While cores are abundant, interconnect is scarce, and no design is best
- OoO Superscalar vs. GPU dichotomy has broken down
- Current ISAs communicate little info to guide energy conservation
- Stored program computers incur high overhead outside of ALUs
Physics & Devices

• **Dennard's scaling: engineering with no tradeoffs**
  - Shrinking transistor increased speed and decreased supply voltage
  - Decreased voltage decreased power consumption

• **Digital limitations**
  - Frequency, voltage scaling now only moderate [6]
  - No blockbuster devices in near future

• **Transistors no longer work as perfect switches**
  - But computation isn't limited to using binary switches—
  - Resistor: multipliers for neural networks
  - Capacitor: integrators for equation solvers
  - Memristor: storage devices
Digital vs. Analog Data

- **Benefit of digital data: high signal-to-noise**
  - Allows abstractions expressed as logic
  - Error correction

- **Motivation for analog representation:**
  - More interaction between cyber & physical
  - Not just as peripherals!
Analog: Logic & Pipelines

Threshold level of C

Spike train from A

Spike train of C

Generator potential in C

Spike train from B
Figure 1. Neural network hierarchy containing convolutional, pooling and classifier layers.
Analog: Organization of Cores

- Analog neural networks support digital computers at various granularities
  - Pipeline level: Neuflow [8], NPUs [26]
  - Standalone accelerator: DianNao
  - Standalone system: IBM TrueNorth [7]
  - Standalone system, integrating main memory: DaDianNao [4]
• Analog datapath & networks are often circuit switched...
  • Program instructions are embodied in datapath topology at runtime [4], [8]

• ...permitting spike train encoding of data on fewer wires...
  • Floating point value is encoded in spike train density [7]
  • Makes operations such as multiplication and integration efficient

• ...and prefers asynchronous continuous time data transmission.
  • When using spike, voltage, or current encodings, there is no notion of clock cycles
Analog: Programs & Workloads

• **Analog requires new programming models**
  • Analog computers lack instruction memory, time mux
  • Program is embodied in datapath setup [8]
  • Solution: compose subnets of neural nets: divide & conquer declarative programming [7]

• **Demonstrated Analog Workloads**
  • Function approximation for floating point programs [26]
  • Neural network pattern recognition for vision / ML [4], [8]
  • …and hardware may drive workloads evolution, too
<table>
<thead>
<tr>
<th><strong>Programming model</strong></th>
<th><strong>Digital</strong></th>
<th><strong>Analog</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Imperative, temporal</td>
<td>Declarative, spatial</td>
</tr>
<tr>
<td><strong>Network on chip</strong></td>
<td>Packet switched</td>
<td>Circuit switched</td>
</tr>
<tr>
<td><strong>Datapath</strong></td>
<td>Synchronous discrete time</td>
<td>Asynchronous continuous time</td>
</tr>
<tr>
<td><strong>Registers</strong></td>
<td>Digital registers</td>
<td>Analog capacitors</td>
</tr>
<tr>
<td><strong>Data representation</strong></td>
<td>Digital integers, IEEE floating point</td>
<td>Analog voltage, current, spike density</td>
</tr>
<tr>
<td><strong>Function units</strong></td>
<td>ALUs</td>
<td>Neurons, etc.</td>
</tr>
<tr>
<td><strong>Physical devices</strong></td>
<td>Relies on ideal transistor</td>
<td>Many physical processes</td>
</tr>
</tbody>
</table>

“Analog computing” changes whole stack, each previously explored; which ones are source of benefit?
$\Theta = -0.5 \times \Theta - 5 \times \sin \Theta, \ \Theta(0) = 1$
HCDC Demonstration

\[
\frac{\text{Analog Input}}{\text{Analog Output}} \rightarrow \frac{\text{DAC}}{-5x} \rightarrow \frac{\sin(x)}{\text{ADC}} \rightarrow \text{Output}
\]
HCDC API Library Calls

34   hcdcInit();
35
36   // Set initial integrator values
37   float initial_y0 = 5.000000;
38
39   // Call HCDC wiring instructions
40   setSimpleConn ( {fans[0], out0, muls[0], in0} );
41   setSimpleConn ( {fans[0], out1, muls[0], in1} );
42   setSimpleConn ( {fans[0], out2, ints[0], in0} );
43
44   setSimpleConn ( {muls[0], out0, ints[0], in1} );
45
46   setIntInitial ( ints[0], initial_y0 );
47   setSimpleConn ( {ints[0], out0, fans[0], in0} );