A different way to formulate computing: Optimal Adiabatic Scaling (OAS) and Processor-In-Memory-and-Storage (PIMS)

Erik P. DeBenedictis
Energy efficiency can depend on clock rate

- David Frank (IBM) discussed adiabatic and reversible computing at RCS 2, where energy efficiency varies by clock rate.

- Adiabatic circuits have behavior close to:
  - Energy/op $\propto f$ (clock rate)
  - Power $\propto f^2$

- This would be equivalent to slope 1 on chart at left.

- This effect depends on:
  - Adiabatic circuitry
  - Devices – 11 nm adiabatic CMOS and nSQUID on David Frank’s chart, but many other options.

- Let’s work with this.

From David Frank’s presentation at RCS 2; viewgraph 23
A plot will reveal what we will call Optimal Adiabatic Scaling (OAS)

- Impact of manufacturing cost
  - At RCS 2, David Frank put forth the idea that a computer costs should include both purchase cost and energy cost.
  - However, let’s adapt this idea to a situation where manufacturing cost drops with time, as in Moore’s Law
- Let’s plot economic quality of a chip:
  \[ Q_{\text{chip}} = \frac{\text{Ops}_{\text{lifetime}}(f)}{\$_{\text{purchase}} + \$_{\text{energy}}(f^2)} \]
  Where \$_{\text{purchase}} = A
  \text{Ops}_{\text{lifetime}} = Bf, and
  \$_{\text{energy}} = Cf^2 \text{ (A, B, and C constants)}

- Assume manufacturing costs drops to \( \frac{1}{2} \) every three years
- Top of ridge rises with time

![Graph showing economic quality of a chip over time with Optimal Adiabatic Scaling (OAS)](image)
Arguably, this scaling rule explains multi-core and the non-emergence of reversible computing

- Prior to around 2003, purchase costs dominated energy
  - The economically enlightened approach would be to raise clock rate, which happened
- Around 2003, technology went over the optimal point
  - Multi-core was the technical remedy to the economic problem – had lower clock rate
- Reversible computing would be an advance in the right direction, but too extreme for now
### Resulting scaling scenario (standard chart with additional column)

If $C$ and $V$ stop scaling, throughput ($f N_{\text{tran}} N_{\text{core}}$) stops scaling.

<table>
<thead>
<tr>
<th></th>
<th>Const field</th>
<th>Constant $V$</th>
<th>Optimal Adiabatic Scaling</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_{\text{gate}}$</td>
<td>$1/\alpha$</td>
<td>$1/\alpha$</td>
<td>$1^*$</td>
</tr>
<tr>
<td>$W, L_{\text{wire}}$</td>
<td>$1/\alpha$</td>
<td>$1/\alpha$</td>
<td>$N=\alpha^2$†</td>
</tr>
<tr>
<td>$V$</td>
<td>$1/\alpha$</td>
<td>$1$</td>
<td>$1$</td>
</tr>
<tr>
<td>$C$</td>
<td>$1/\alpha$</td>
<td>$1/\alpha$</td>
<td>$1/\alpha$</td>
</tr>
<tr>
<td>$U_{\text{stor}} = \frac{1}{2} CV^2$</td>
<td>$1/\alpha^3$</td>
<td>$1/\alpha$</td>
<td>$1/\sqrt{N}=1/\alpha$²‡</td>
</tr>
<tr>
<td>$f$</td>
<td>$\alpha$</td>
<td>$\alpha$</td>
<td>$1/\sqrt{N}=1/\alpha$</td>
</tr>
<tr>
<td>$N_{\text{tran}}/\text{core}$</td>
<td>$\alpha^2$</td>
<td>$\alpha^2$</td>
<td>$1$</td>
</tr>
<tr>
<td>$N_{\text{core}}/A$</td>
<td>$1$</td>
<td>$1$</td>
<td>$\sqrt{N}=\alpha$</td>
</tr>
<tr>
<td>$P_{\text{ckt}}$</td>
<td>$1/\alpha^2$</td>
<td>$1/\alpha$</td>
<td>$1/\sqrt{N}=1/\alpha$</td>
</tr>
<tr>
<td>$P/A$</td>
<td>$1$</td>
<td>$\alpha^2$</td>
<td>$1$</td>
</tr>
<tr>
<td>$f N_{\text{tran}} N_{\text{core}}$</td>
<td>$\alpha^3$</td>
<td>$\alpha^3$</td>
<td>$\sqrt{N}=\alpha$</td>
</tr>
</tbody>
</table>

* Term redefined to be line width scaling; 1 means no line width scaling
† Term redefined to be the increase in number of layers; previously was 1 for no scaling
‡ Term redefined to be heat produced per step. Adiabatic technologies do not reduce signal energy, but “recycle” signal energy so the amount turned into heat scales down
§ Term clarified to be power per unit area including all devices stacked in 3D

Ref: T. Theis, In Quest of the “Next Switch”: Prospects for Greatly Reduced Power Dissipation in a Successor to the Silicon Field-Effect Transistor, Proceedings of the IEEE, Volume 98, Issue 12, 2010

Under OAS, throughput continues to scale even with fixed $V$ and $C$.
Physical implementation in 3D

- Same device behavior
  - If there are improvements in device behavior, they create an improvement over and above what is illustrated
- Exponentially improving manufacturing cost (Moore’s Law)

True 3D manufacture
100 nm³ gates
$10^{15}$ gates
300× increase in power efficiency
300× increase in throughput
~10 MHz

100 nm² gates

From RCS 2
Need a new architecture; von Neumann architecture won’t do

- OAS scales throughput
  - Device count scales up by $N (N = \alpha^2)$
  - Clock rate scales down by $1/\sqrt{N}$
  - Throughput scales up by $N \times 1/\sqrt{N} = \sqrt{N}$

- The von Neumann architecture cannot exploit this throughput
  - Processor and memory contribute independently to performance
  - Slower computer with more memory – not viable

- We need an architecture whose performance is the product of memory size and clock rate
  - Processor-in-memory?
    - Easily said, but we need a specific architecture that scales properly and has good generality
Processor-In-Memory-and-Storage (PIMS)

- We class this as an “ALU on column” “processor-in-memory” (PIM) architecture, with persistent storage
  - We use PIM as a descriptive phrase, but it is often used as a name for their specific architecture (GilgaMesh, DIVA, etc.)
- Example chip (one layer of stack):

  ![Diagram](image)

  - Chip is 128×128 array of above

- Architecture characteristics
  - Like a storage-augmented systolic array
  - Must be adiabatically clocked, which is mainly a constraint on the memory
  - Replication unit described as GPU--

Equivalent density to 128 gb Flash
Potential physical implementation

- Storage/Memory
  - Flash, ReRAM (memristor), STM, DRAM
- Base layer
  - PIMS logic
- 3D
  - Whole structure is layered
- External processor?
  - Might be needed for applications without sufficient parallelism
  - Might be needed for programmers who don’t want to recode
  - More on this later
Adiabatic memory

Energy-recycling row drive of a memory:

Result: $85\times$ energy efficiency improvement:

![Diagram of adiabatic memory](image)

**Source:**

1.1 TMACS/mW Fine-Grained Stochastic Resonant Charge-Recycling Array Processor

Rafał Karakiewicz, Senior Member, IEEE, Roman Genov, Member, IEEE, and Gert Cauwenberghs, Fellow, IEEE
What applications scale like PIMS?

- We already decided it would not make good components for a von Neumann machine
- However, PIMS scales like an overall computer system
  - “Kryder’s law” reveals that disk storage has grown at about twice the rate of Moore’s Law
- PIMS also scales like a brain
  - Example scale up sequence
    - roundworm, fruit fly, honeybee, mouse, rat, human
  - Brain = robot controller function
- Scales like a parallel supercomputer, but not like an individual node
PIMS example: sparse matrix for neural networks, Deep Learning, etc.

- Neural networks frequently compute as sparse matrices
  - Vector-matrix multiply
  - Delta learning rule
    - matrix += vector outer product
- Efficiency example loads sparse matrix at 45° angle

- Architecture encodes sparse matrix structure in memory/storage array
- Permits MIMD PIM operation with high power efficiency
  - Apparently novel

Memory array

Logic

Rearrange-ment registers
Exemplary ALU

- Note that this is neither a microprocessor nor a GPU

Storage array format:

<table>
<thead>
<tr>
<th>Synapse value: 8 bits as signed integer, but often interpreted at a higher level as a fixed point number</th>
<th>Green pointer code word</th>
<th>Red pointer code word</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 bits +</td>
<td>2 bits +</td>
<td>2 bits</td>
</tr>
</tbody>
</table>

12 bits total:

ALU (one for each 12 storage bits):
Performance on Deep Learning example

- Scale to human brain size of $10^{11}$ neurons and $10^{15}$ synapses
- Energy subdivides into two components
  - Memory access energy (energy per bit $\times$ bits)
    - Options: non-adiabatic DRAM PIM, adiabatic memory, NVIDIA GTX 750 Ti
  - Synapse evaluation energy (depends on number of bits precision)
    - Options: TFET and extrapolated CMOS, NVIDIA GTX 750 Ti
- Result
  - Non-adiabatic DRAM about $2000 \times$ more energy efficient than GPU
  - Additional $50 \times$ more efficient with adiabatic memory
## Performance on Deep Learning example

<table>
<thead>
<tr>
<th>Logic</th>
<th>Mem-ory</th>
<th>DRAM</th>
<th>TMACS</th>
<th>nVidia GTX 750 Ti</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>fj/synapse</td>
<td>bits needed</td>
<td>46 fj/bit</td>
<td>9.1 fj/bit</td>
</tr>
<tr>
<td>TFET</td>
<td>1.3</td>
<td>12</td>
<td>$12 \times 46 = 552$ (fj memory)</td>
<td>$12 \times 9.1 = 111$ (fj memory)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$1.3$ (fj logic)</td>
<td>$1.3$ (fj logic)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$553$ (fj mem+logic)</td>
<td>$12$ (fj mem+logic)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$11$ KW (kilowatts)</td>
<td>$240$ W</td>
</tr>
<tr>
<td>HP</td>
<td>21.8</td>
<td>12</td>
<td>$12 \times 46 = 552$ (fj memory)</td>
<td>$12 \times 9.1 = 111$ (fj memory)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$21.8$ (fj logic)</td>
<td>$21.8$ (fj logic)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$574$ (fj mem+logic)</td>
<td>$33$ (fj mem+logic)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$11$ KW</td>
<td>$650$ W</td>
</tr>
<tr>
<td>TFET</td>
<td>7.7</td>
<td>21</td>
<td>$21 \times 46 = 1149$ (fj memory)</td>
<td>$21 \times 9.1 = 23$ (fj memory)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$7.7$ (fj logic)</td>
<td>$7.7$ (fj logic)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$1158$ (fj mem+logic)</td>
<td>$30$ (fj mem+logic)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$23$ KW</td>
<td>$610$ W</td>
</tr>
<tr>
<td>HP 21</td>
<td>128</td>
<td>21</td>
<td>$21 \times 46 = 1149$ (fj memory)</td>
<td>$21 \times 9.1 = 23$ (fj memory)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$128$ (fj logic)</td>
<td>$128$ (fj logic)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$1278$ (fj mem+logic)</td>
<td>$150$ (fj mem+logic)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$26$ KW</td>
<td>$3$ KW</td>
</tr>
</tbody>
</table>

**Legend:**
- **Line 1:** femto joules to access memory for one synapse
- **Line 2:** femto joules logic energy to act on a synapse
- **Line 3:** Total energy (line 1 + line 2)
- **Line 4:** System energy at specified scale (watts, kilowatts, megawatts)
Conclusions

There is plenty of room for continued improvement

- **Device physics**
  - Dimensions are small enough and signal energy is low enough
  - Now work in the third dimension and recycle signal energy

- **Architecture**
  - Microprocessor and memory are just components and they must be connected by a non-scalable bus
  - So build the system instead of the components and leave out the bus

- **Software**
  - We program via large numbers of manipulations of small data types, which requires physically unrealistic clock rate scaling
  - Instead, develop larger primitives and program them at a higher level
    - Use sparse matrices, neurons, etc. as the primitives
    - Like 3D graphics on a GPU
Data model for Processor-In-Memory-and-Storage (PIMS)

A. von Neumann model with input/output:

Read input
Parse
Process with $\sqrt{N}$ efficiency boost
Format
Write output

B. Processor-In-Memory-and-Storage:

Read input
Parse
Process with $\sqrt{N}$ efficiency boost
Format
Write output

C. Persistent object store of data in form for optimal access:

Read input
Parse
Process with $\sqrt{N}$ efficiency boost
Format
Write output