Scaling Beyond Moore's Law with Processor-In-Memory-and-Storage (PIMS)

Erik P. DeBenedictis

ITRS ERD/IEEE Rebooting Computing Meeting

February 26, 2015

Approved for Unclassified Unlimited Release
SAND2015-1333 PE

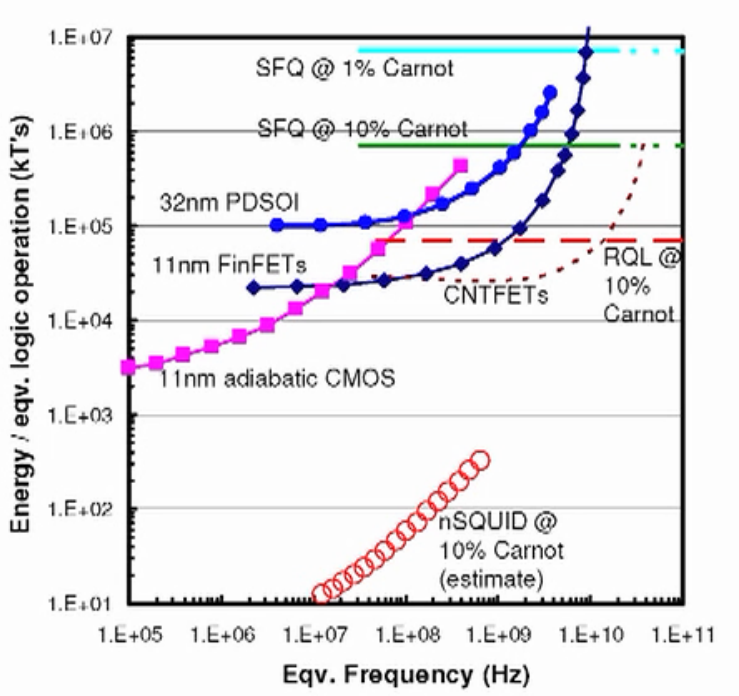
Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Outline

- Formulate 3D scaling rule
- Architecture options
  - ☒ von Neumann
  - ☑ Logic-memory integration
- Programming
- Performance
- Device implications

Energy efficiency can depend on clock rate

- David Frank (IBM) studied energy efficiency variance by clock rate
- Can make a scaling rule out of *f* vs energy efficiency dependence?



From David Frank's presentation at RCS 2; viewgraph 23. "Yes, I'm ok with the viewgraphs being public, so it's ok for you to use the figure. Dave" (10/31/14)

- 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

A plot will reveal what we will call "optimal adiabatic scaling"

- 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 gate or chip:

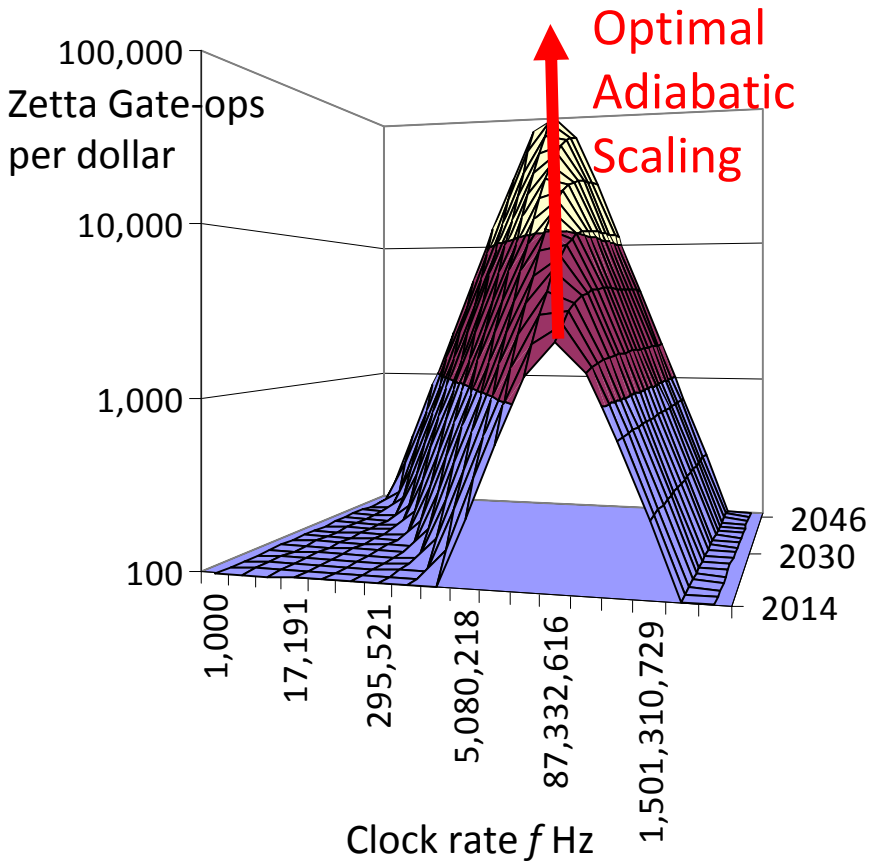
$$Q_{chip} = \frac{Ops_{lifetime}(f)}{\$_{purchase} + \$_{energy}(f^2)}$$

Where $\$_{purchase} = A\ 2^{-t_{year}/3}$

$Ops_{lifetime} = Bf$, *and*

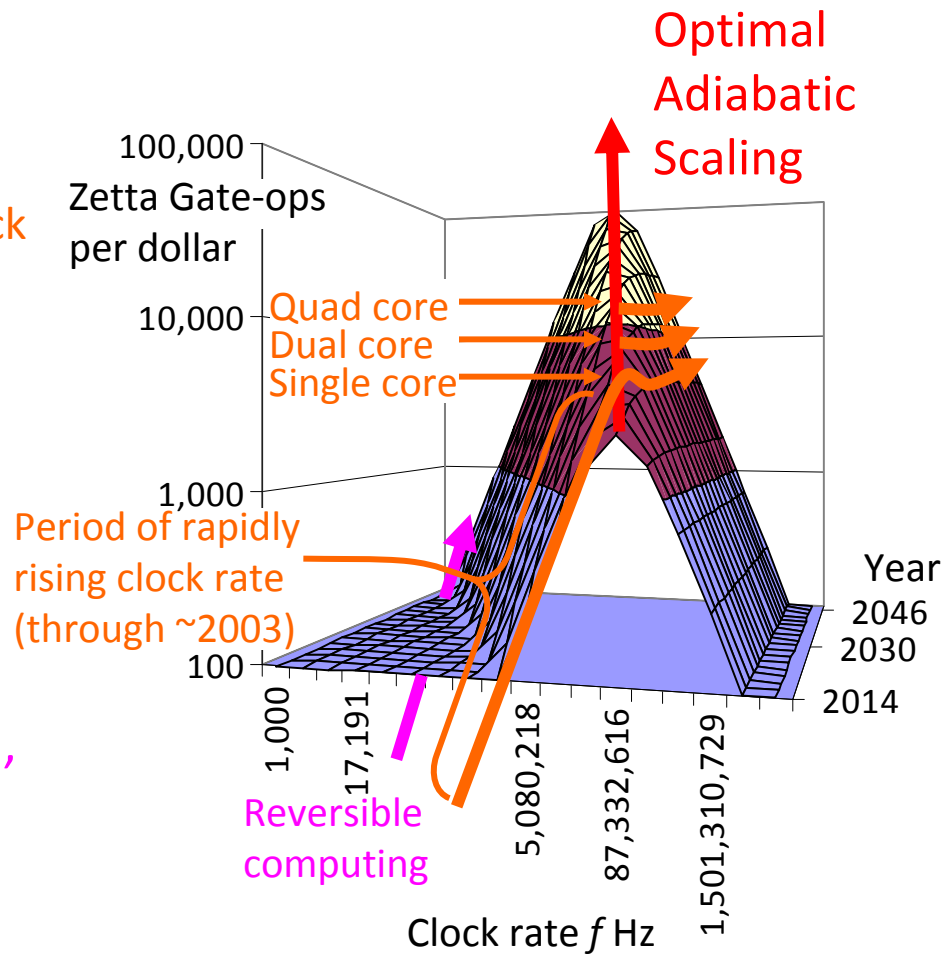
$\$_{energy} = Cf^2$ (A, B, and C constants)

- Assume manufacturing costs drops to ½ every three years
- Top of the ridge rises with time



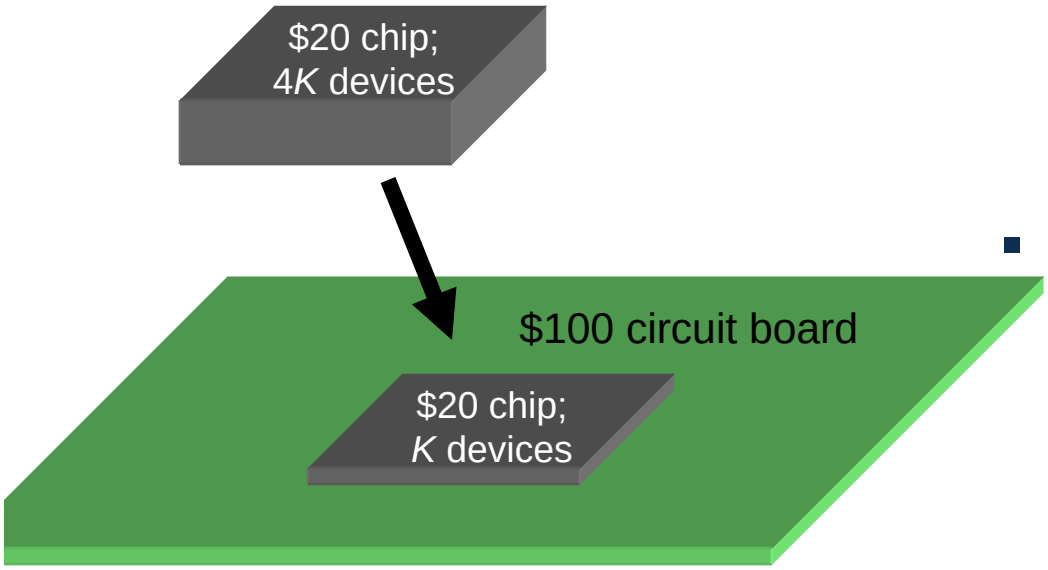
Backup: historical context and reversible computing

- 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



How to derive a scaling rule

- Chip vendor says: "How would you like a chip with 4× as many devices for the same price?"



- Optimal adiabatic scaling says:
  - Cut clock rate to $1/\sqrt{4}\times$ (halve)
  - Power per device drops to 1/4×
  - Power per chip stays same
  - Throughput doubles: 4× as many devices runn at $1/\sqrt{4}\times$ the speed, for a net throughput increase of $\sqrt{4}\times$
- "Throughput" is in accordance with the way throughput is measured for semiconductors, which does not include effects of architecture and algorithms (which we discuss later)
- To make a scaling rule, replace "4" with α^2 (line width scaling)

Resulting scaling scenario (standard chart with additional column)

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

	Const field	Constant V				Optimal Adiabatic Scaling
		Max f	Const f	Const f, N_{tran}	Multi core	
L_{gate}	$1/\alpha$	$1/\alpha$	$1/\alpha$	$1/\alpha$	$1/\alpha$	1*
W, L_{wire}	$1/\alpha$	$1/\alpha$	$1/\alpha$	1	$1/\alpha$	$N=\alpha^2$†
V	$1/\alpha$	1	1	1	1	1
C	$1/\alpha$	$1/\alpha$	$1/\alpha$	1	$1/\alpha$	1
$U_{stor} = ½\ CV^2$	$1/\alpha^3$	$1/\alpha$	$1/\alpha$	1	$1/\alpha$	$1/\sqrt{N}=1/\alpha$‡
f	α	α	1	1	1	$1/\sqrt{N}=1/\alpha$
N_{tran}/core	α^2	α^2	α^2	1	1	1
N_{core}/A	1	1	1	1	α	$\sqrt{N}=\alpha$
P_{ckt}	$1/\alpha^2$	1	$1/\alpha$	1	$1/\alpha$	$1/\sqrt{N}=1/\alpha$
P/A	1	α^2	α	1	1	1§
$f\ N_{tran}\ N_{core}$	α^3	α^3	α^2	1	α	$\sqrt{N}=\alpha$

Theis and Solomon → New

Under optimal adiabatic scaling, throughput continues to scale even with fixed V and C

* 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

Outline

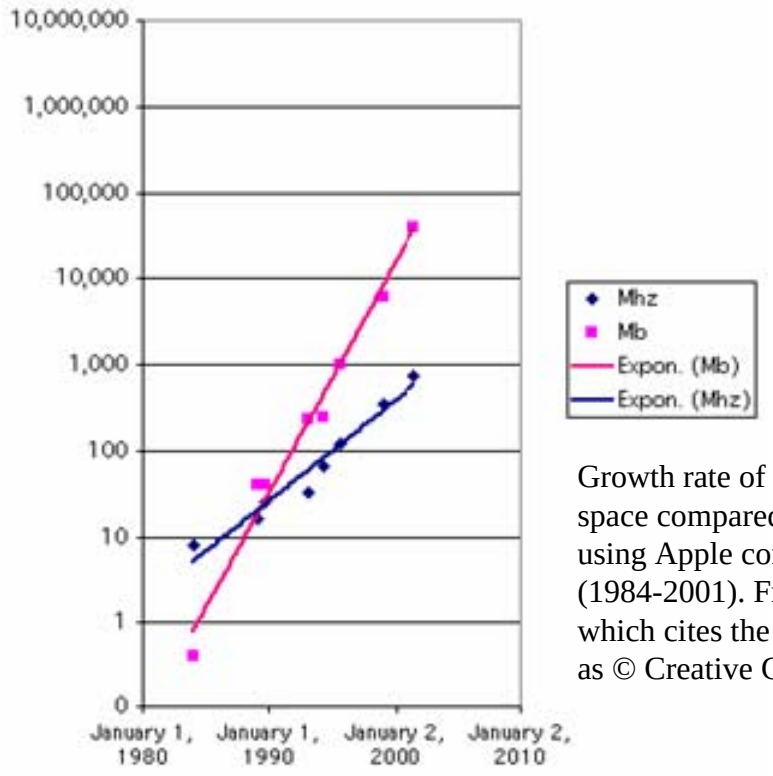
- Formulate 3D scaling rule
- Architecture options
  - ☒ von Neumann
  - ☑ Logic-memory integration
- Programming
- Performance
- Device implications

Need a new architecture; von Neumann architecture won't do

- Optimal adiabatic scaling proportions
  - 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

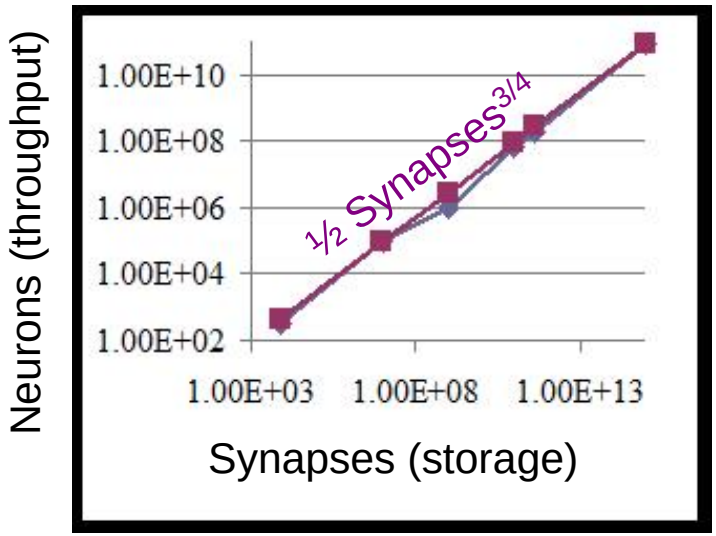
What applications scale like PIMS?

- Computer system clock rate grew at about the square root the rate of storage capacity



Growth rate of HDD storage space compared to clock rate using Apple consumer products (1984-2001). From Wikipedia, which cites the diagram to left as © Creative Commons.

- Brain CPU throughput grows at ¾ power of storage capacity
  - Which is consistent because brains get bigger too



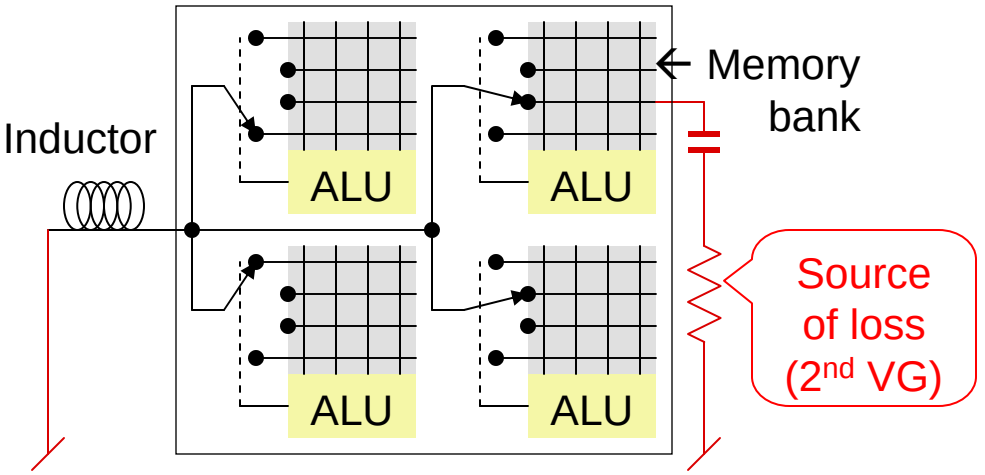
	Synapses	Neurons
Roundworm	7.50E+03	3.02E+02
Fruit fly	1.00E+07	1.00E+05
Honeybee	1.00E+09	9.60E+05
Mouse	1.00E+11	7.10E+07
Rat	4.48E+11	2.00E+08
Human	1.00E+15	8.60E+10

Source: Wikipedia

Design for energy management

- Design around fixing competitor's weakest features:
  - Von Neumann bus/bottleneck
  - CV^2 losses
- Make principal energy pathway into a resonant circuit
  - Recycle the energy that the competitor's system turns into heat

- Chip



- Size expectations for 128 Gb
  - 1024×1024 bits/memory bank
  - 128×128 banks/chip

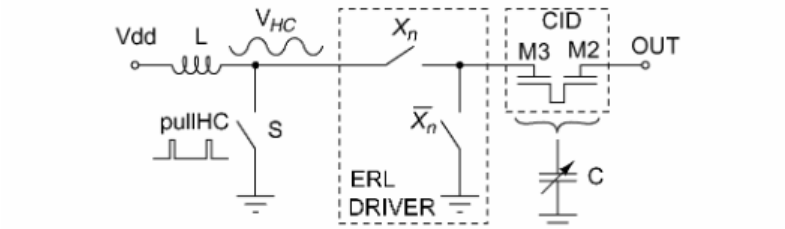
Backup: adiabatic memory (low) maturity level

- Source

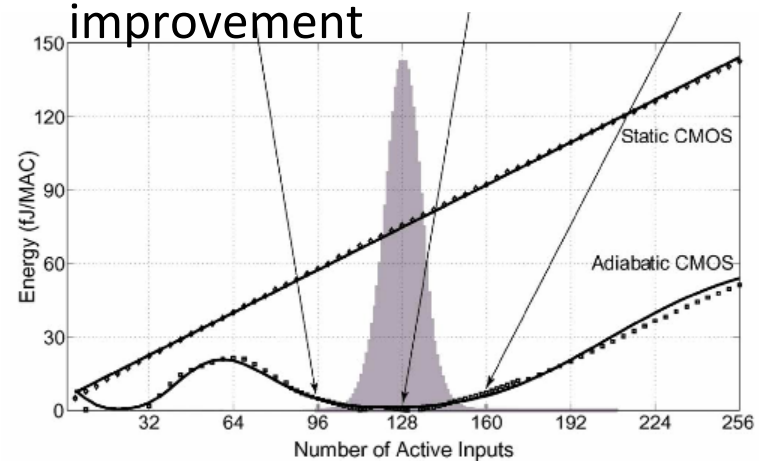
1.1 TMACS/mW Fine-Grained Stochastic Resonant Charge-Recycling Array Processor

Rafal Karakiewicz, *Senior Member, IEEE*, Roman Genov, *Member, IEEE*, and Gert Cauwenberghs, *Fellow, IEEE*

- Energy-recycling row drive



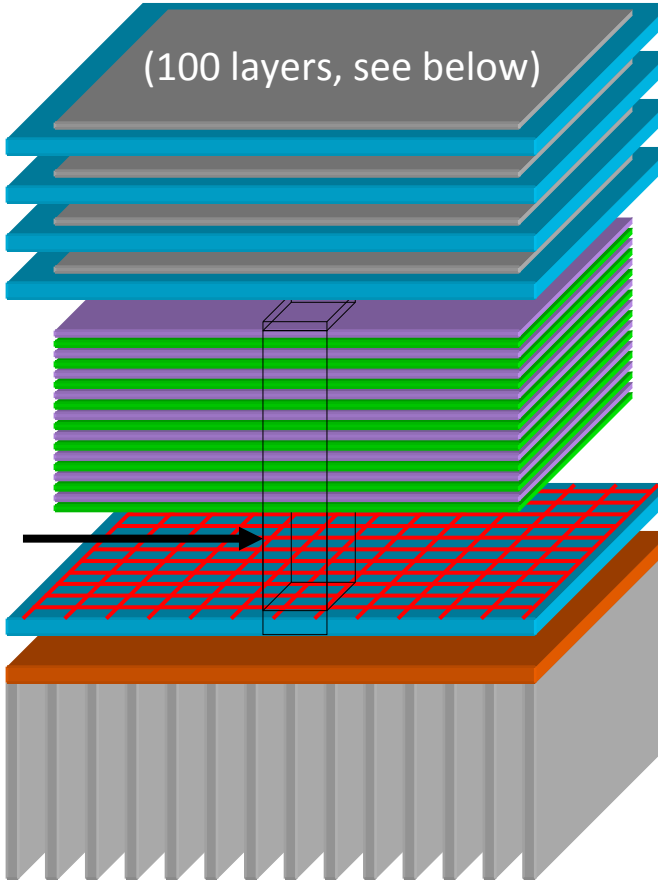
- Result 85× energy efficiency improvement



- TRL 3 or 4 for Charge Injection Devices (CID). TRL definitions:
  - 3. Analytical and experimental critical function and/or characteristic proof of concept
  - 4. Component and/or breadboard validation in laboratory environment
- Above research is for charge injection devices. Author does not see a theoretical reason why it could not work for memristors and flash
- Resonators and inductors ought to be OK

Nominal physical implementation

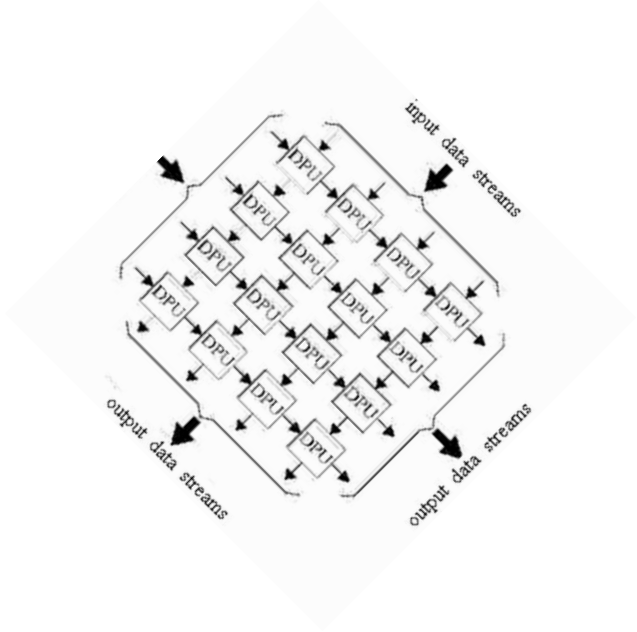
- Storage/Memory
  - Flash, ReRAM (memristor), STM, DRAM
- Base layer
  - PIMS logic
- 3D
  - Whole structure is layered
- SOME ADDITIONAL DETAIL IN BACKUP



Outline

- Formulate 3D scaling rule
- Architecture options
  - ☒ von Neumann
  - ☑ Logic-memory integration
- Programming
- Performance
- Device implications

Capturing the scaling power efficiency with programming



x			
1	2	3	4

A			
1	0	0	2
0	0	3	0
0	4	0	5
6	0	0	0

=

y			
25	12	6	17

Vector-matrix multiply on left implemented by dataflow-like spreadsheet below.

Timestep 1:

Timestep 2:

Etc.

				x_0 1				y_0 0							
		x_1 2			a00 1 x_0 1 y_0 1				y_1 0						
	x_2 3			a10 0 x_1 2 y_0 1			a01 0 x_0 1 y_1 0					y_2 0			
x_3 4			a20 0 x_2 3 y_0 1			a11 0 x_1 2 y_1 0				a02 0 x_0 1 y_2 0				y_3 0	
	a30 6 x_3 4 y_0 25			a21 4 x_2 3 y_1 12			a12 3 x_1 2 y_2 6			a03 2 x_0 1 y_3 2					
y_0 25		a31 0 x_3 4 y_1 12			a22 0 x_2 3 y_2 6			a13 0 x_1 2 y_3 2							
	y_1 12			a32 0 x_3 4 y_2 6			a23 5 x_2 3 y_3 17								
		y_2 6			a33 0 x_3 4 y_3 17										
				y_3 17											

Note: the y_j's are updated, so they do not all have the same value

1st cell column above, as it evolves with time

2nd cell column above, as it evolves with time

3rd cell, and so on

Note on above: this diagram is only a spreadsheet, but you may think of a row of x's and y's as a register that shifts right and left each time step; the a's do not shift (see arrows).

Tile programming

x

1	2	3	4

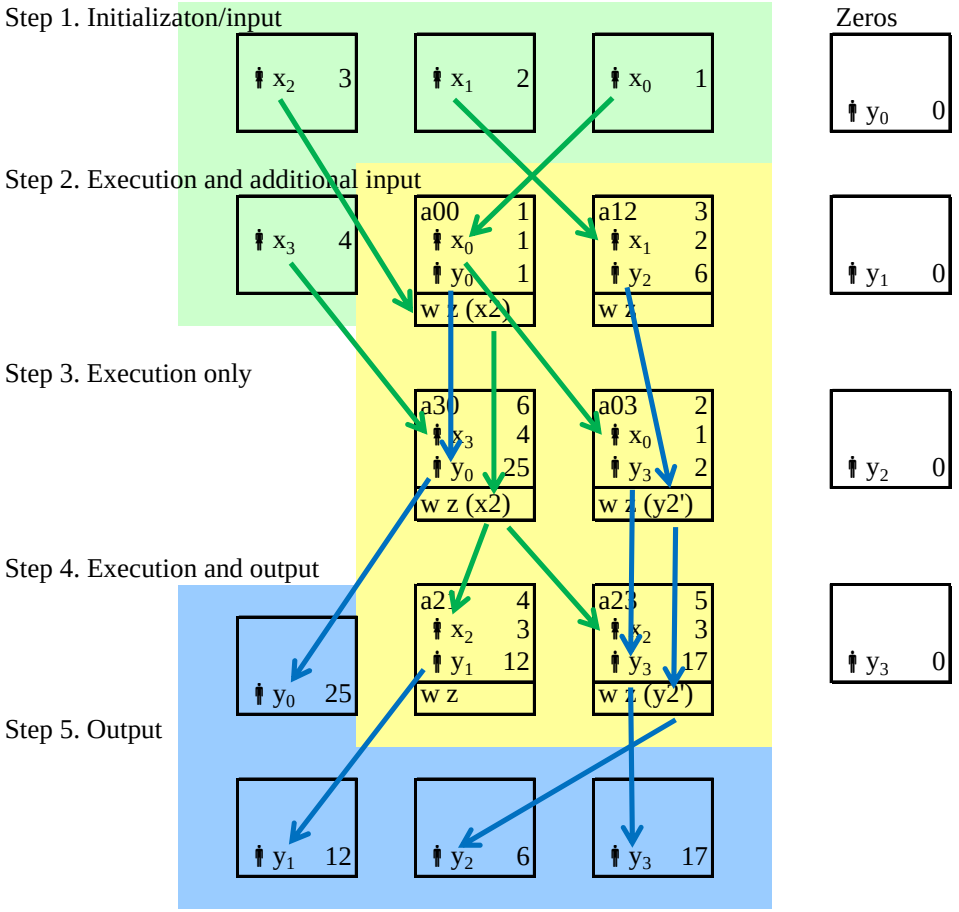
A

1			2
		3	
	4		5
6			

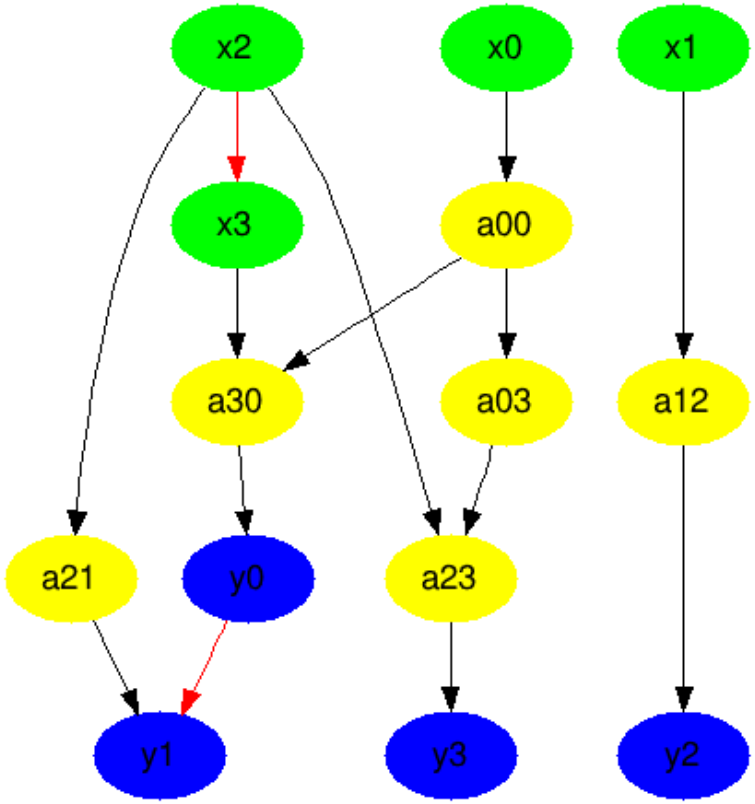
= y

25	12	6	17

Arrows indicate data flow; wth no data flow faster than nearest neighbor per step. Sometimes dance steps for ladies and gents.



GraphViz:



Outline

- Formulate 3D scaling rule
- Architecture options
  - ☒ von Neumann
  - ☑ Logic-memory integration
- Programming
- Performance
- Device implications

Baseline performance expectations

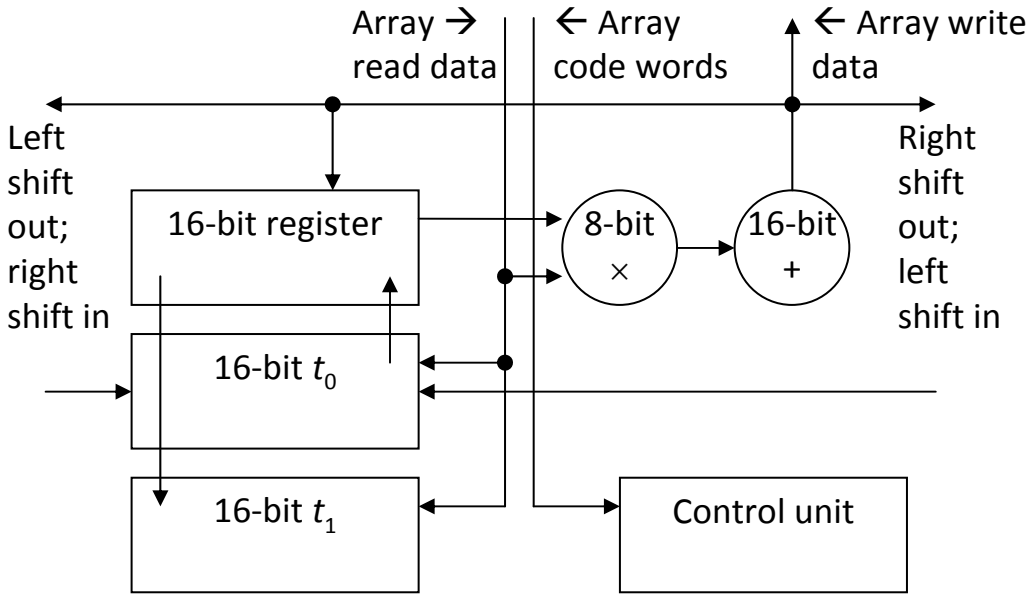
- Application: Deep learning on 10^{11} neurons 10^{15} synapses

Test case ALU

Storage array format:

Synapse value: 8 bits as signed integer, but often interpreted at a higher level as a fixed point number	Green pointer code word	Red pointer code word
12 bits total: 8 bits +	2 bits +	2 bits

ALU (one for each 12 storage bits):



Performance on Deep Learning example

Note: NVIDIA GTX 750 Ti is memory bandwidth limited so the logic energy is ignored.

CMOS HP and TFET per Nikonov and Young's study

First two rows are 8-bit synapse; last two rows are 16-bit synapse

Memory		GTX 750 Ti		DRAM		Adiabatic Mem	
		0.1	nJ/bit	46.0	fJ/bit	0.9	fJ/bit
Logic type							
TFET		1.0	nJ	552.0	fJ	10.9	fJ
1.3	fj/synapse	0.0	J	1.3	fJ	1.3	fJ
12	bits needed	1.0	nJ	553.3	fJ	12.2	fJ
		20.8	MW	11.1	KW	244.3	W
CMOS HP		1.0	nJ	552.0	fJ	10.9	fJ
21.8	fj/synapse	0.0	J	21.8	fJ	21.8	fJ
12	bits needed	1.0	nJ	573.7	fJ	32.7	fJ
		20.8	MW	11.5	KW	653.2	W
TFET 21 bits		2.2	nJ	1150.0	fJ	22.7	fJ
7.7	fj/synapse	0.0	J	7.7	fJ	7.7	fJ
25	bits needed	2.2	nJ	1157.6	fJ	30.4	fJ
		43.4	MW	23.2	KW	607.9	W
CMOS HP 21 bits		2.2	nJ	1150.0	fJ	22.7	fJ
127.8	fj/synapse	0.0	J	127.8	fJ	127.8	fJ
25	bits needed	2.2	nJ	1277.7	fJ	150.5	fJ
		43.4	MW	25.6	KW	3010.2	W
Line 1: Femto joules to access memory for one synapse							
Line 2: Femto joules logic energy to act on one synapse							
Line 3: Sum of previous two lines							
Line 4: System energy (watts, kilowatts, megawatts)							

Baseline commentary:

80,000× energy efficiency boost – quite a bit

Memory energy still dominates, even with adiabatic memory

Outline

- Formulate 3D scaling rule
- Architecture options
  - ☒ von Neumann
  - ☑ Logic-memory integration
- Programming
- Performance
- Device implications

Device implications; conclusions

Device implications

- There is nothing wrong with transistor function
- We need to drive down manufacturing cost, which probably requires a new device
  - could be a more manufacturable transistor
  - could be something different, but the difference is not essential
- Logic-memory integration is essential

Conclusions

- With logic-memory integration, we could possibly have an exponential improvement path until we end up with a structure with the parameters of a brain (throughput/storage)
  - We don't claim to know how to program a brain
- This VG deck is showing how to turn lower manufacturing cost into higher device efficiency

Are there three neuromorphic options (besides software)?

- Software (Deep Learning, etc.)
- Crossbar with a boost from level-based analog (memristor)
- Spiking with a boost from time-based analog
- Digital emulation of neurons with a boost from adiabatic digital tricks and 3D integration

Expected comparison result

- We did a study of energy efficiency of neuromorphic approaches
- Not ready for publication (too hard)
- Conclusions
  - Physical limits of computation apply to both analog and digital
  - Scale, coding, sparsity, precision determine winner