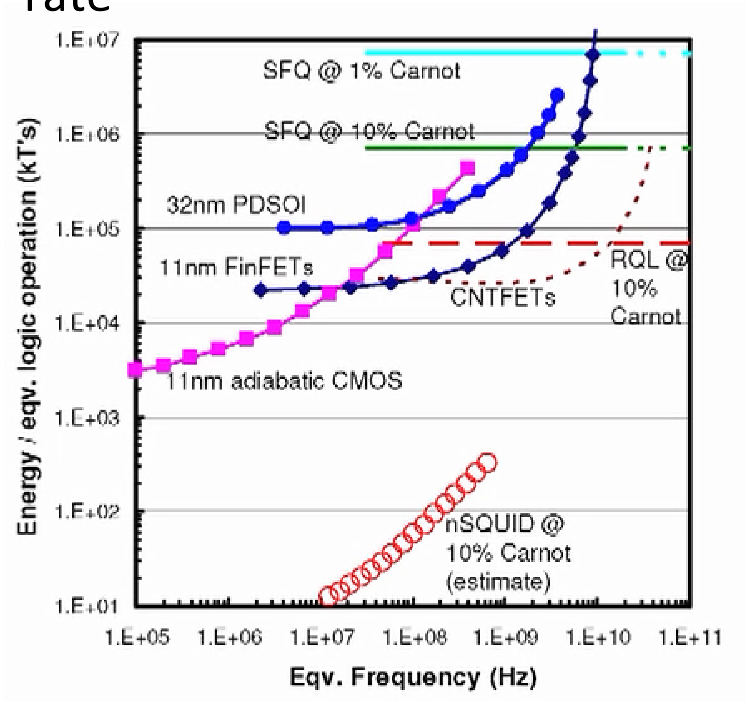
A different way to formulate computing: Optimal Adiabatic Scaling (OAS) and Processor-In-Memory-and-Storage (PIMS)

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



From David Frank's presentation at RCS 2; viewgraph 23
https://ieeetv.ieee.org/player/html/viewer?dl=#-reversible-adiabatic-classical-computation-an-overview-rebooting-computing

- 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 (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:

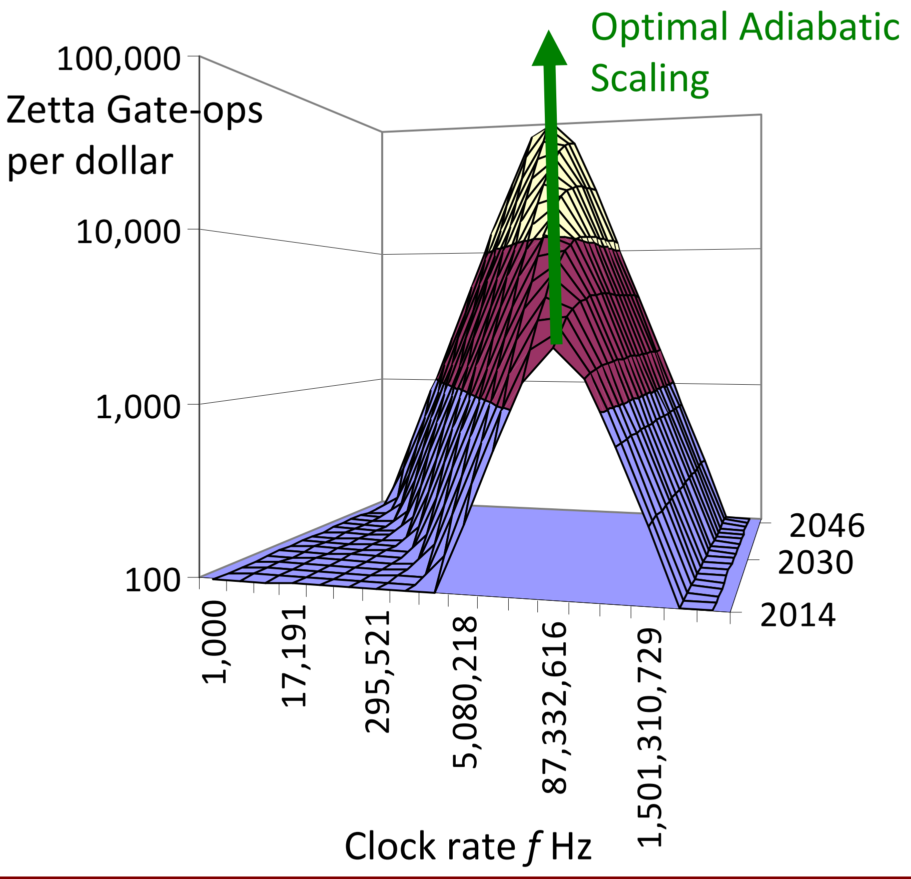
$$Q_{chip} = \frac{Ops_{lifetime}(f)}{\$_{purchase} + \$_{energy}(f^{2})}$$

Where $\$_{purchase} = A$

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

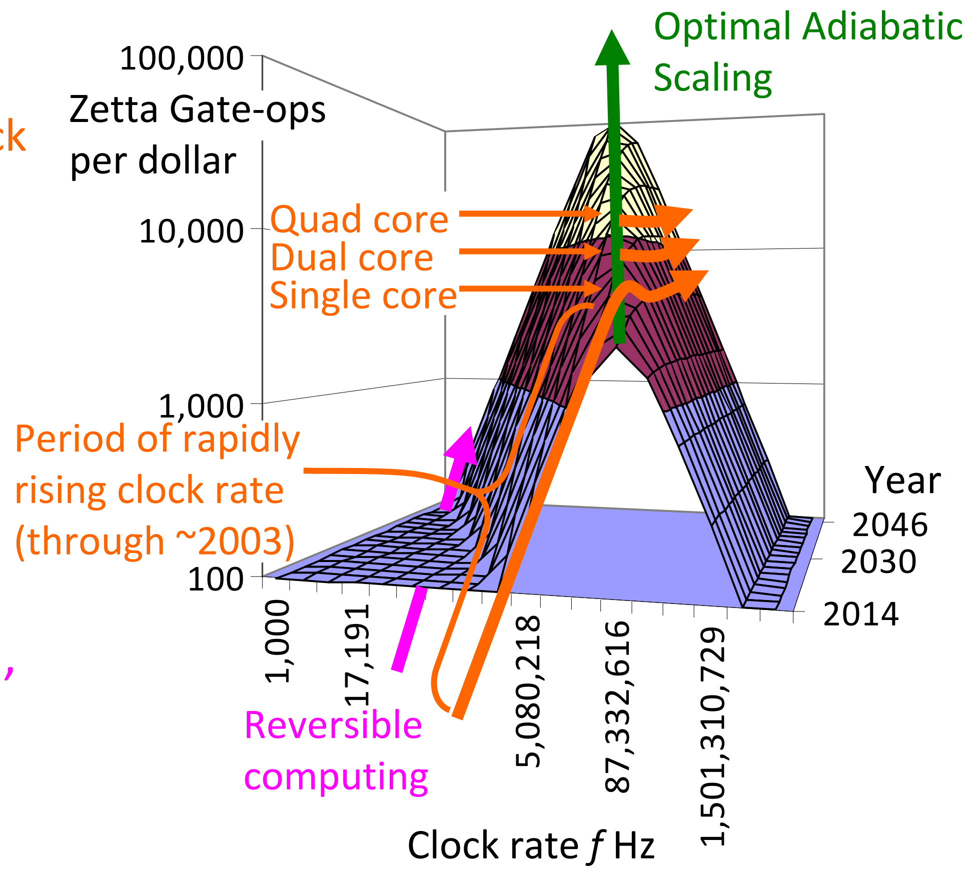
$\$_{energy} = Cf^{2}$ (A, B, and C constants)

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



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 ($f\ N_{tran}\ N_{core}$) stops scaling.

Under OAS, throughput continues to scale even with fixed *V* and *C*

	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	$1/\alpha$	$N=\alpha^{2\dagger}$
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	$1/\alpha$	$1/\sqrt{N}=1/\alpha^{\ddagger}$
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	$1^{\S}$
$f\ N_{tran}\ N_{core}$	α^3	α^3	α^2	1	α	$\sqrt{N}=\alpha$

← Theis and Solomon → New

[*] 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

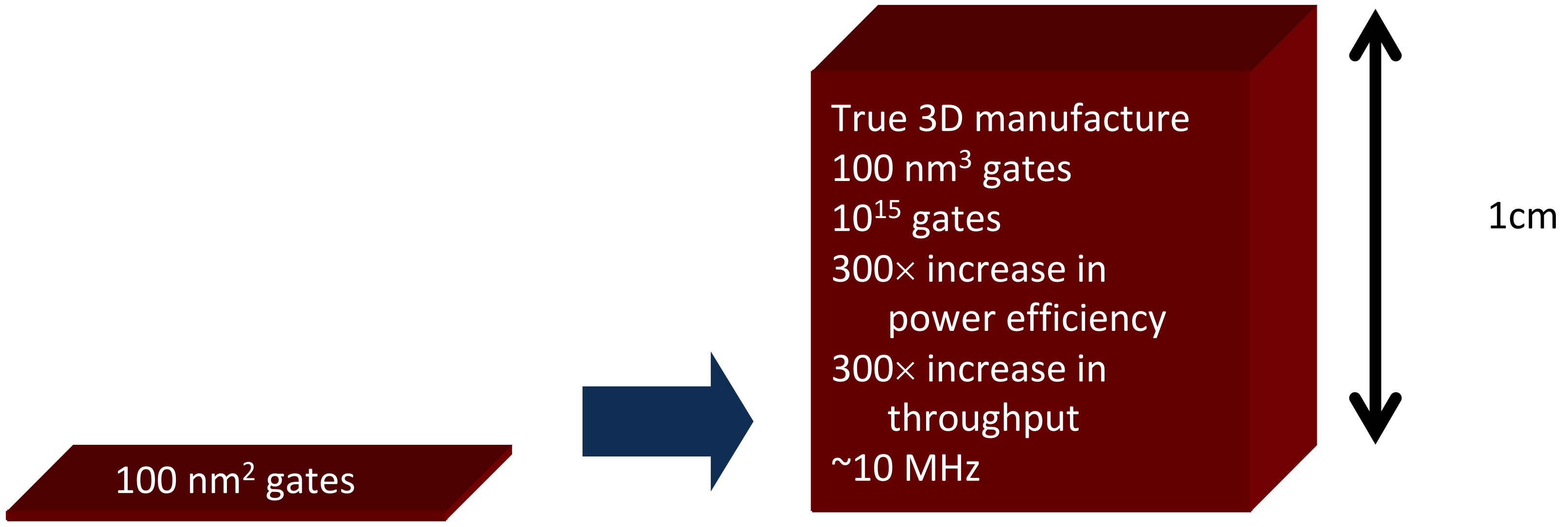
[‡] Term redefined to be heat produced per step. Adiabatic technologies do not reduce signal energy, but "recycle" signal energy so the amount turned into heat scales down

[§] 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

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)



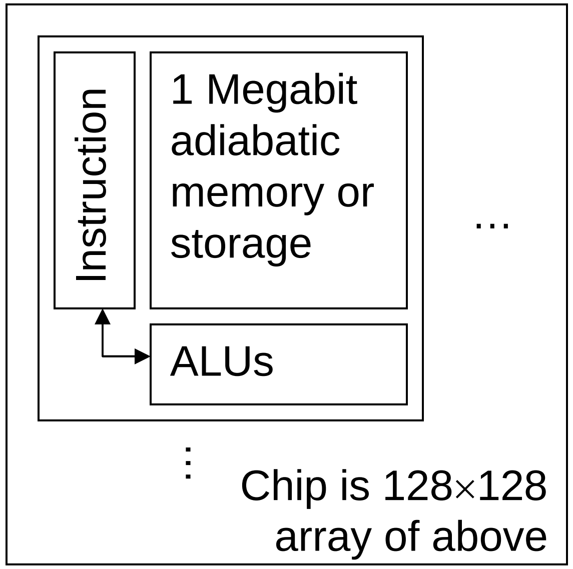
From RCS 2

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):

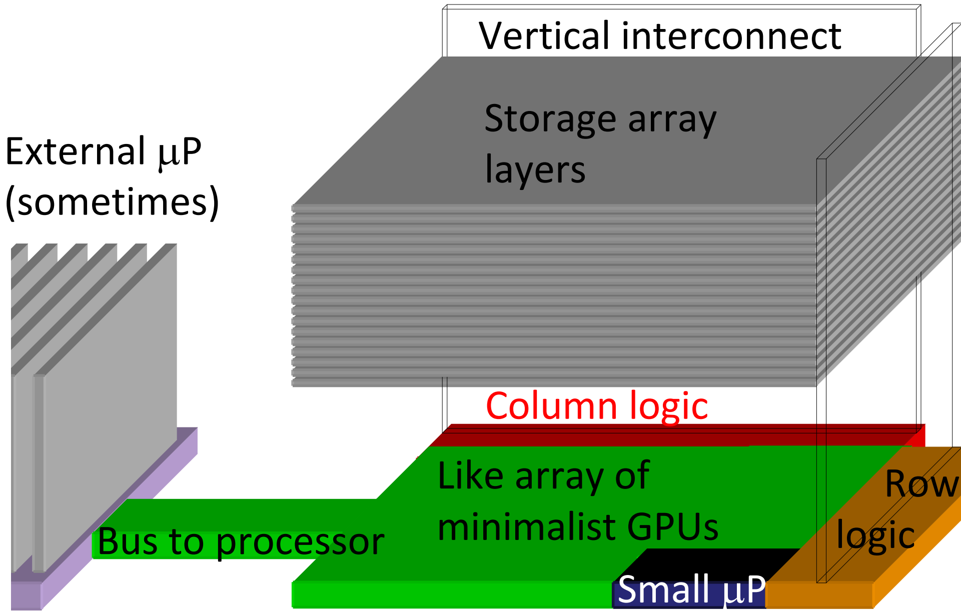

Instruction
1 Megabit adiabatic memory or storage
ALUs
…
⋮
Chip is 128×128 array of above

Equivalent density to 128 gb Flash

- 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--

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:

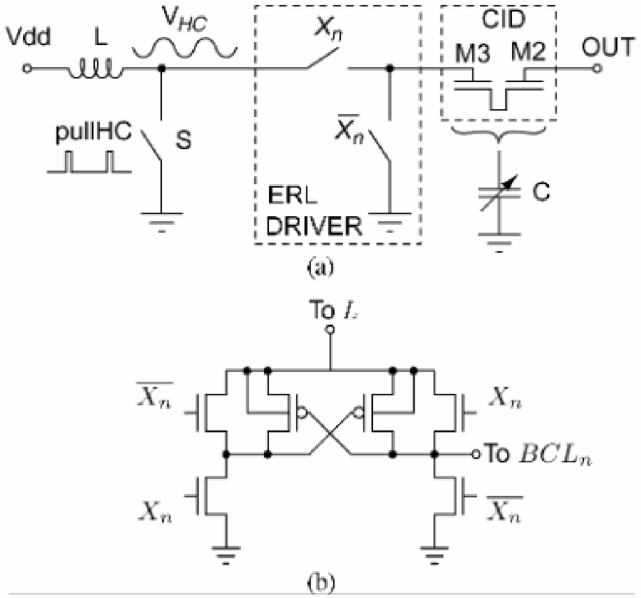


Fig. 7. (a) LC tank oscillator driving an array of charge-recycling CID cells. One cell is shown, with two charge-coupled MOS transistors M2 and M3 according to Figs. 1 and 3. (b) Double-range input-enabled energy recovery logic (ERL) driver.

Result: 85× energy efficiency improvement:

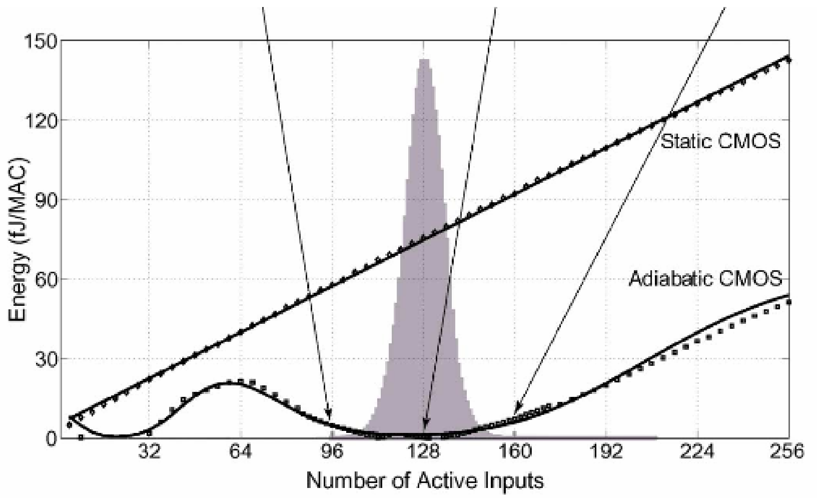


Fig. 12. Experimentally measured (dotted line) and theoretical (solid line) array dynamic energy dissipation as a function of input data statistics in adiabatic resonant mode and static CMOS mode. Three corresponding experimentally measured hot clock waveforms are shown. The probability density function of modulated input data is shown in gray.

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*

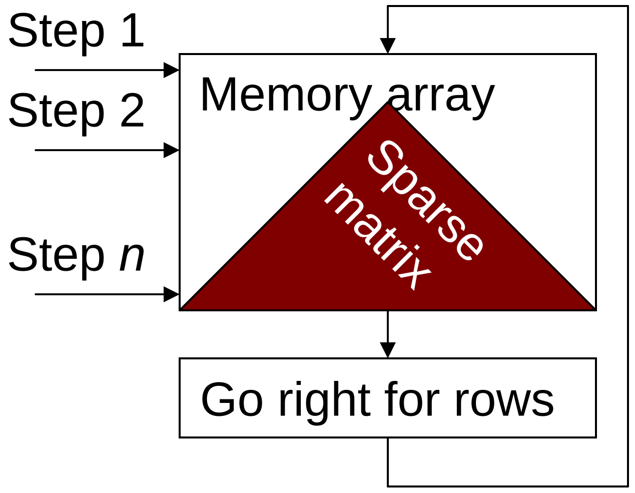
What applications scale like PIMS?

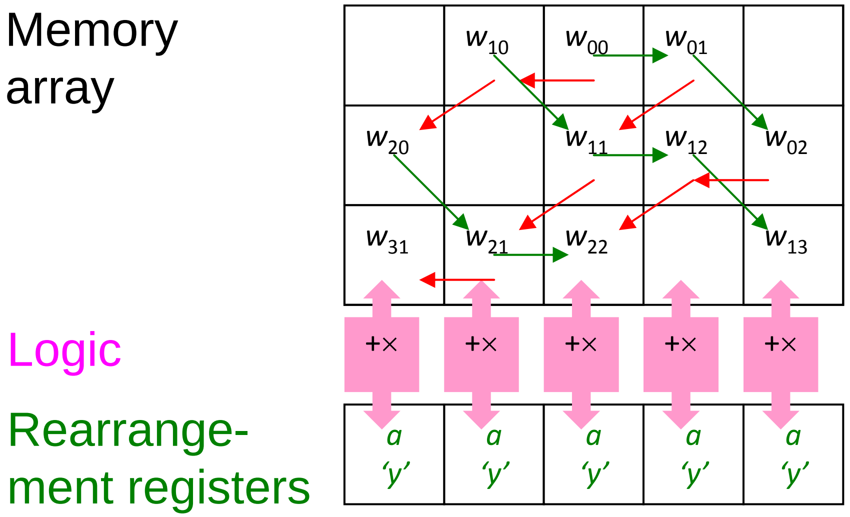
- We already decided it would not make good components for a von Neumann machine
- However, PIMS scales like an overall computer system
  - "Kryder's law" reveals that disk storage has grown at about twice the rate of Moore's Law
- PIMS also scales like a brain
  - Example scale up sequence
    - roundworm, 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





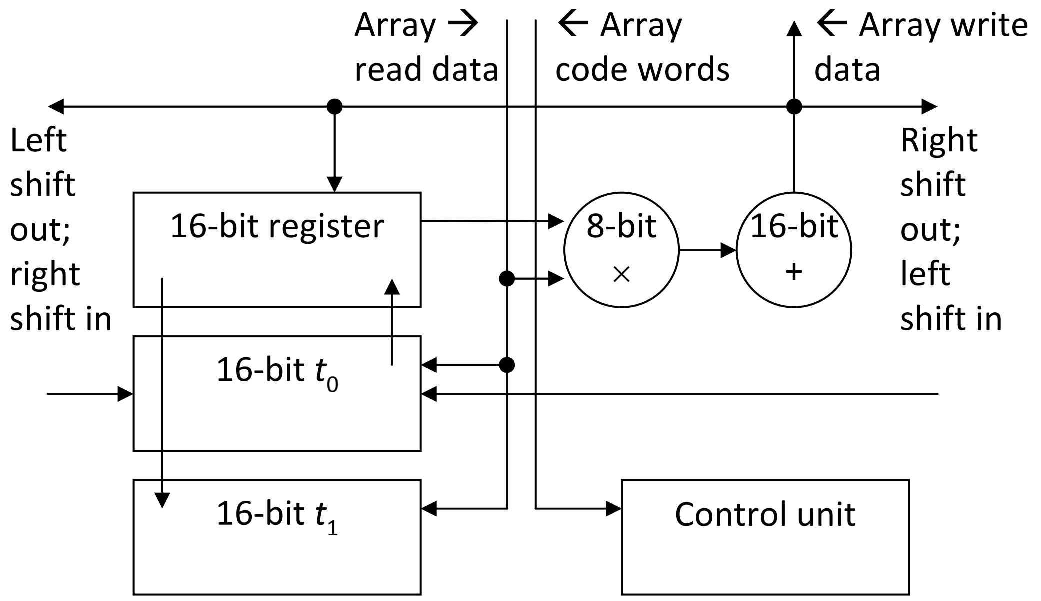
Exemplary ALU

- Note that this is neither a microprocessor nor a GPU

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

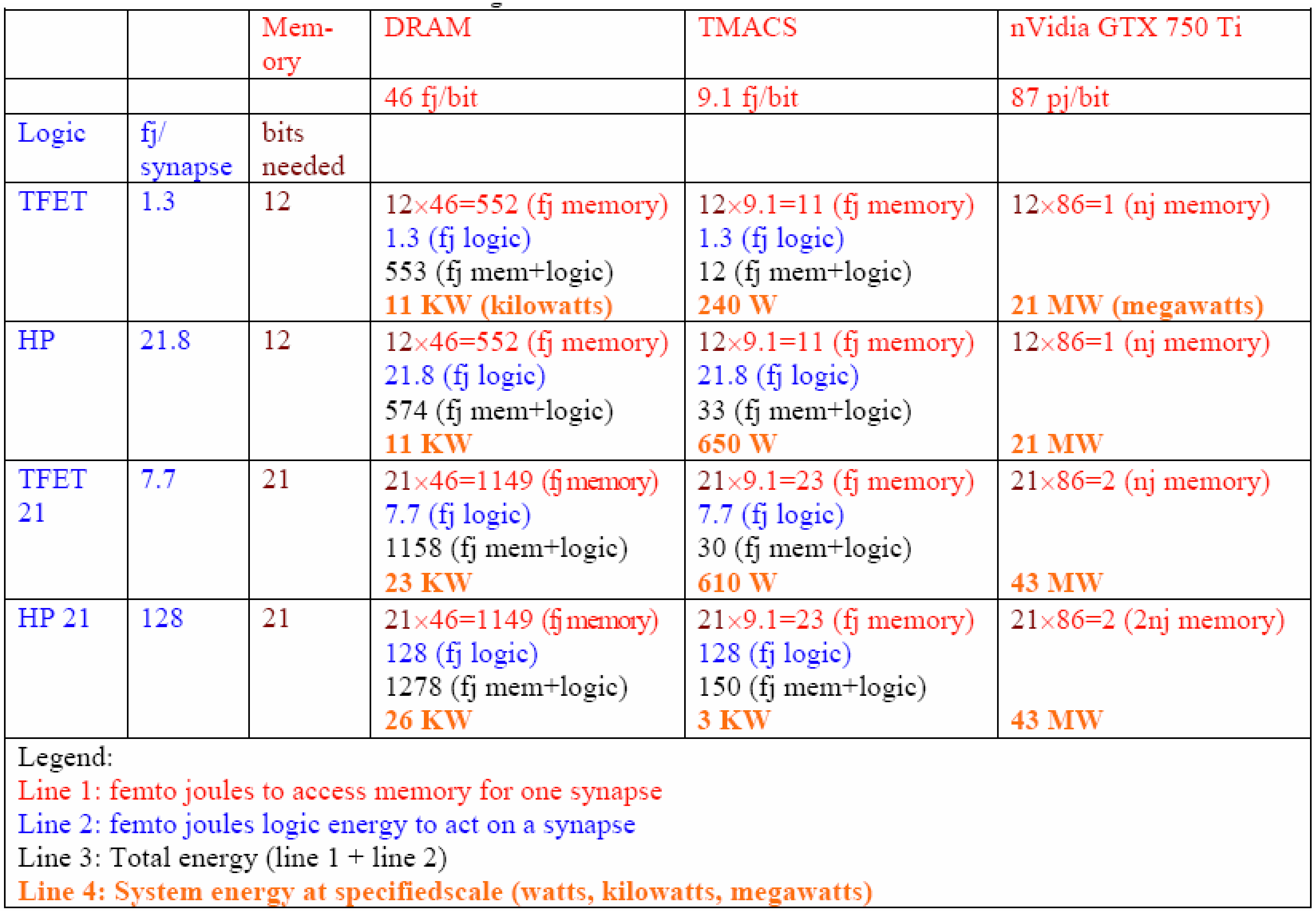
ALU (one for each 12 storage bits):



Performance on Deep Learning example

- Scale to human brain size of 10^{11} neurons and 10^{15} synapses
- Energy subdivides into two components
  - Memory access energy (energy per bit × 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× more energy efficient than GPU
  - Additional 50× more efficient with adiabatic memory

Performance on Deep Learning example



		Mem-ory	DRAM	TMACS	nVidia GTX 750 Ti
			46 fj/bit	9.1 fj/bit	87 pj/bit
Logic	fj/ synapse	bits needed			
TFET	1.3	12	12×46=552 (fj memory) 1.3 (fj logic) 553 (fj mem+logic) **11 KW (kilowatts)**	12×9.1=11 (fj memory) 1.3 (fj logic) 12 (fj mem+logic) **240 W**	12×86=1 (nj memory) **21 MW (megawatts)**
HP	21.8	12	12×46=552 (fj memory) 21.8 (fj logic) 574 (fj mem+logic) **11 KW**	12×9.1=11 (fj memory) 21.8 (fj logic) 33 (fj mem+logic) **650 W**	12×86=1 (nj memory) **21 MW**
TFET 21	7.7	21	21×46=1149 (fj memory) 7.7 (fj logic) 1158 (fj mem+logic) **23 KW**	21×9.1=23 (fj memory) 7.7 (fj logic) 30 (fj mem+logic) **610 W**	21×86=2 (nj memory) **43 MW**
HP 21	128	21	21×46=1149 (fj memory) 128 (fj logic) 1278 (fj mem+logic) **26 KW**	21×9.1=23 (fj memory) 128 (fj logic) 150 (fj mem+logic) **3 KW**	21×86=2 (2nj memory) **43 MW**

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 specifiedscale (watts, kilowatts, megawatts)

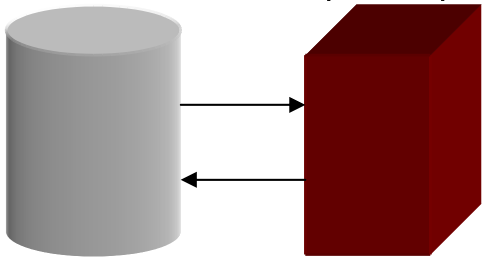
Conclusions

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

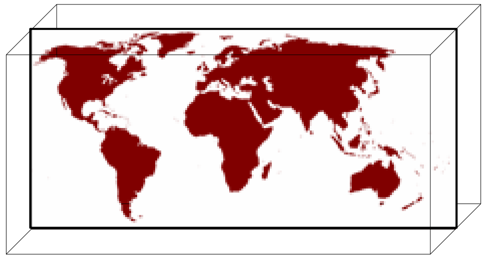
Data model for Processor-In-Memory-and-Storage (PIMS)

A. von Neumann model with input/output:



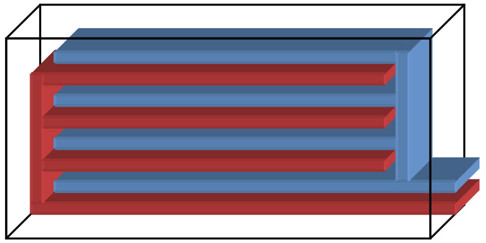
Read input
Parse
Process with √N efficiency boost
Format
Write output

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



~~Read input~~
Parse
Process with √N efficiency boost
Format
~~Write output~~

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



~~Read input~~
~~Parse~~
Process with √N efficiency boost
~~Format~~
~~Write output~~