SAND2014-19930PE

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

Erik P. DeBenedictis

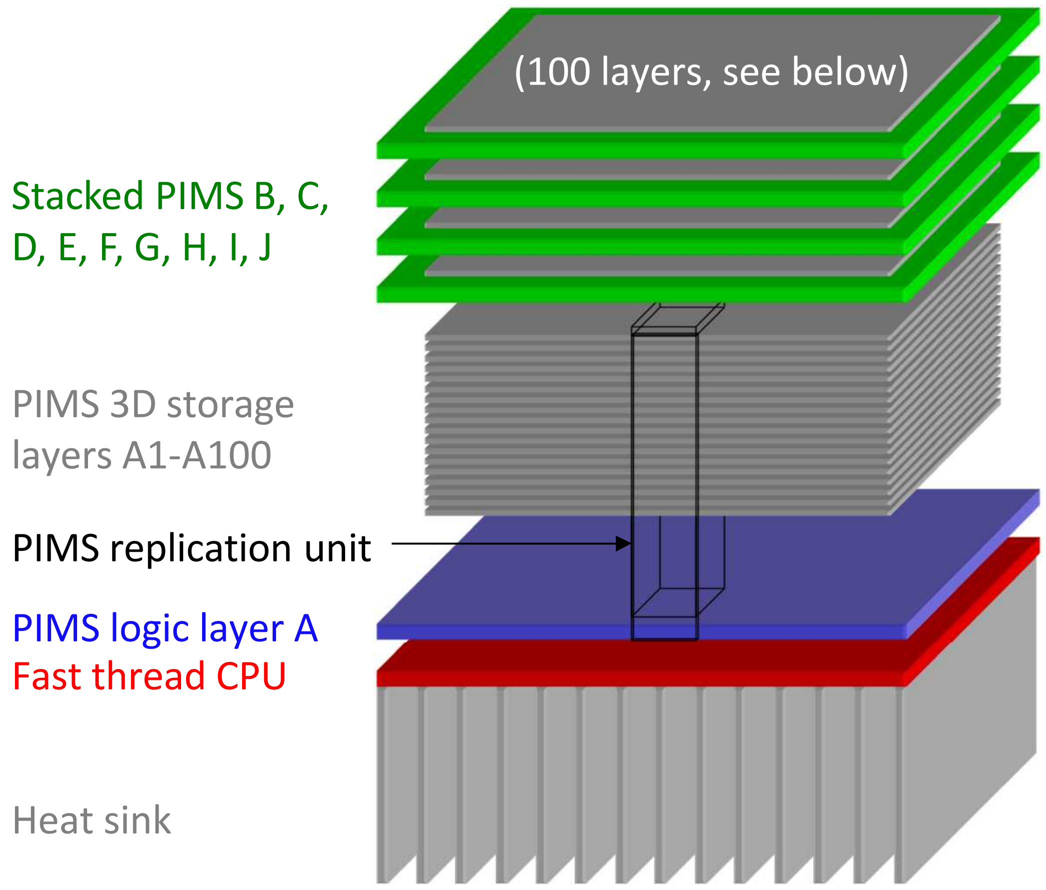
What I will do in this talk

- Describe a plot line for a story
  - "With imagination, the progress of Moore's Law can continue"
- Tell a specific story consistent with the plot
  - Optimal adiabatic scaling +
  - 3D manufacturing +
  - Processor-In-Memory-and-Storage (PIMS) +
  - Deep Learning applications example =
  - Example of Beyond Moore's Law computing
- Challenges to audience
  - Make your own story consistent with the plot line
- Rumors of relevance to evolving events ??

*** PREVIEW ***

	Fast CPU	Gen 1	Gen N
Clock	3 GHz	100 MHz	10 MHz
Devices	10^{10}	10^{13}	10^{15}
Stack × Layers	1 × 1	10 × 100	Molecular assembly?
Ops/joule	1×	30×	300×
Fast thread penalty	.1		
Parallelism boost		3000	30,000
Total throughput	1×	30,000×	300,000×
Power	100W	100W	100W

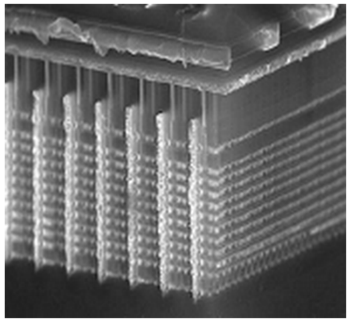
Exploded view:



Backup: stacking ≠ layering & end of Moore's Law

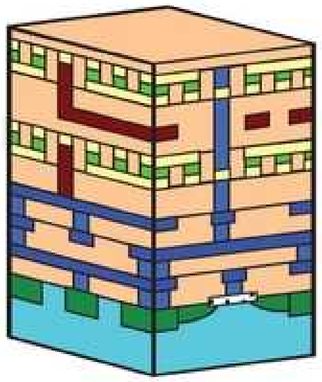
Layering adds additional layers of devices during processing

- Samsung V-NAND



http://www.pcper.com/reviews/Storage/Samsung-850-Pro-512GB-Full-Review-NAND-Goes-3D

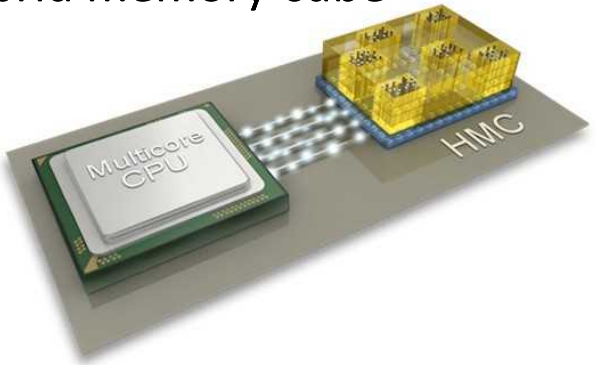
- HP Memristor



Nature

Stacking connects completed chips with Through-Silicon-Vias (TSVs) in an additional processing step

- Hybrid memory cube



http://www.engadget.com/2013/04/03/hybrid-memory-cube-receives-its-finished-spec/

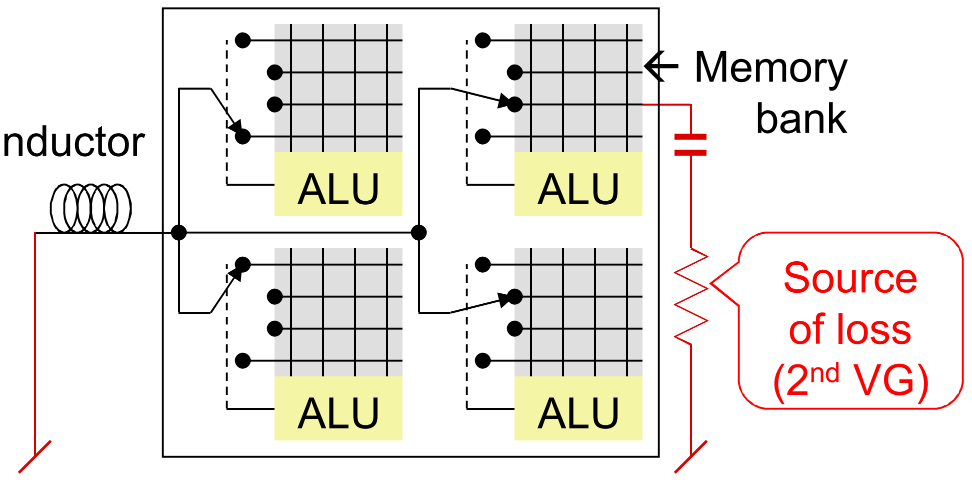
- Disagreement on end of Moore's Law
  - Some say it ended because of 2D feature limits reaching quantum scale
  - Others exploiting third dimension

Outline

- Preview
- Improving power efficiency without changing devices
- Architecture
- Programming
- Performance analysis of example
- Computer system model with integrated I/O

Design for energy management

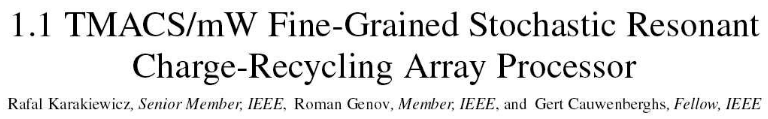
- Design around fixing competitor's weakest features:
  - Von Neumann bus/bottleneck
  - CV^2 losses
- Make principal energy pathway into a resonant circuit
  - Recycle the energy that the competitor's system turns into heat
- Chip



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

Backup: adiabatic memory (low) maturity level

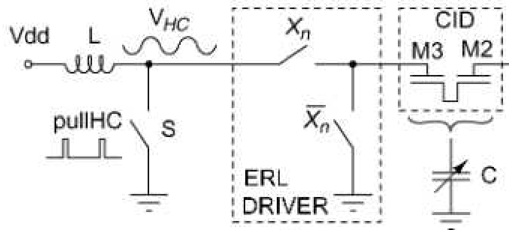
- Source



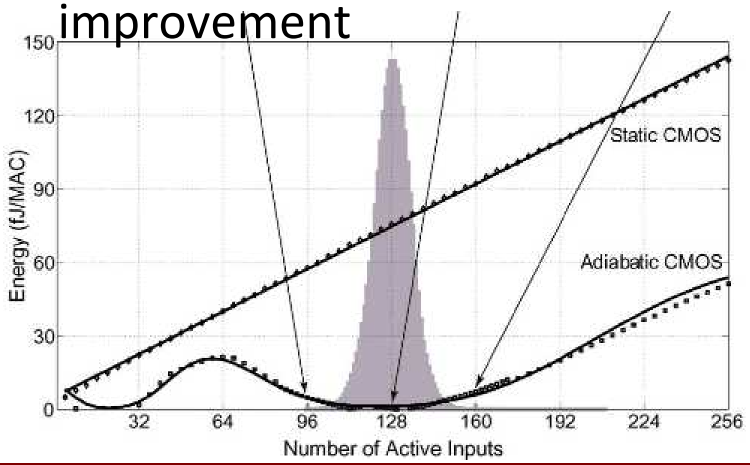
1.1 TMACS/mW Fine-Grained Stochastic Resonant Charge-Recycling Array Processor

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

- Energy-recycling row drive



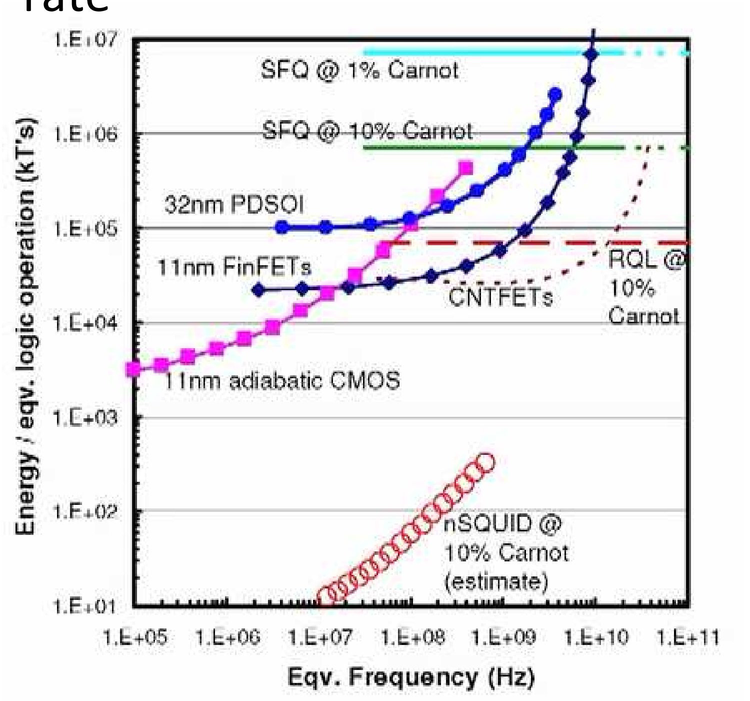
- Result 85× energy efficiency improvement



- 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

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. "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 chip:

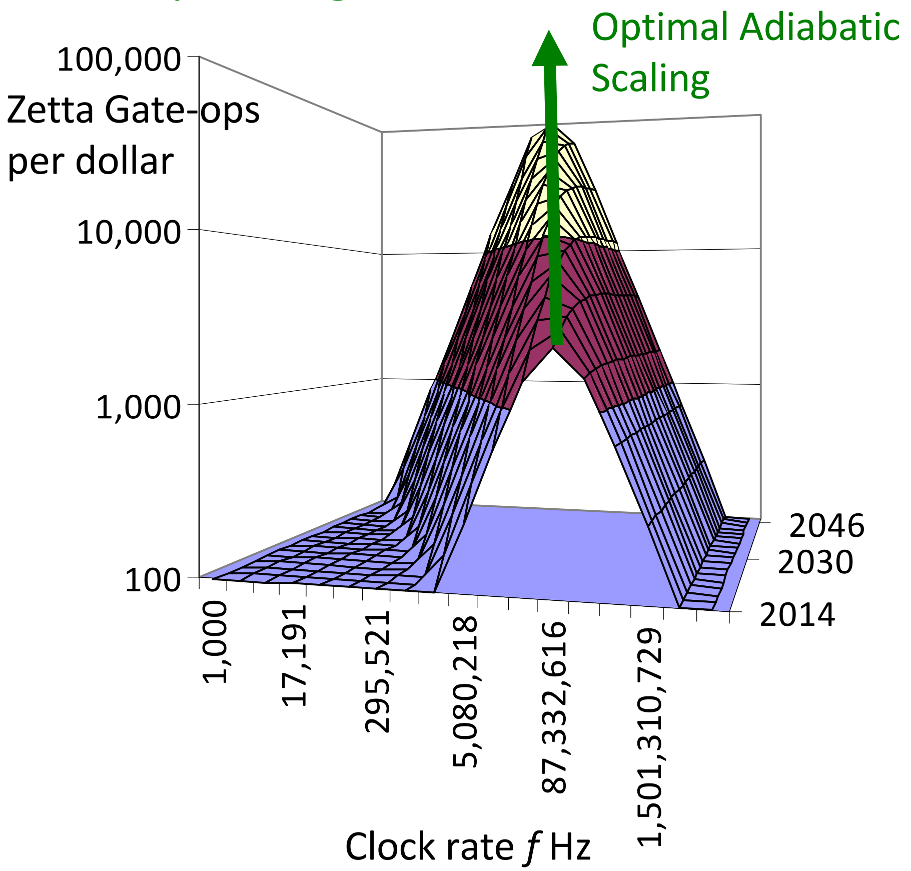
$$Q_{chip} = \frac{Ops_{lifetime}(f)}{\$_{purchase} + \$_{energy}(f^2)}$$

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

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

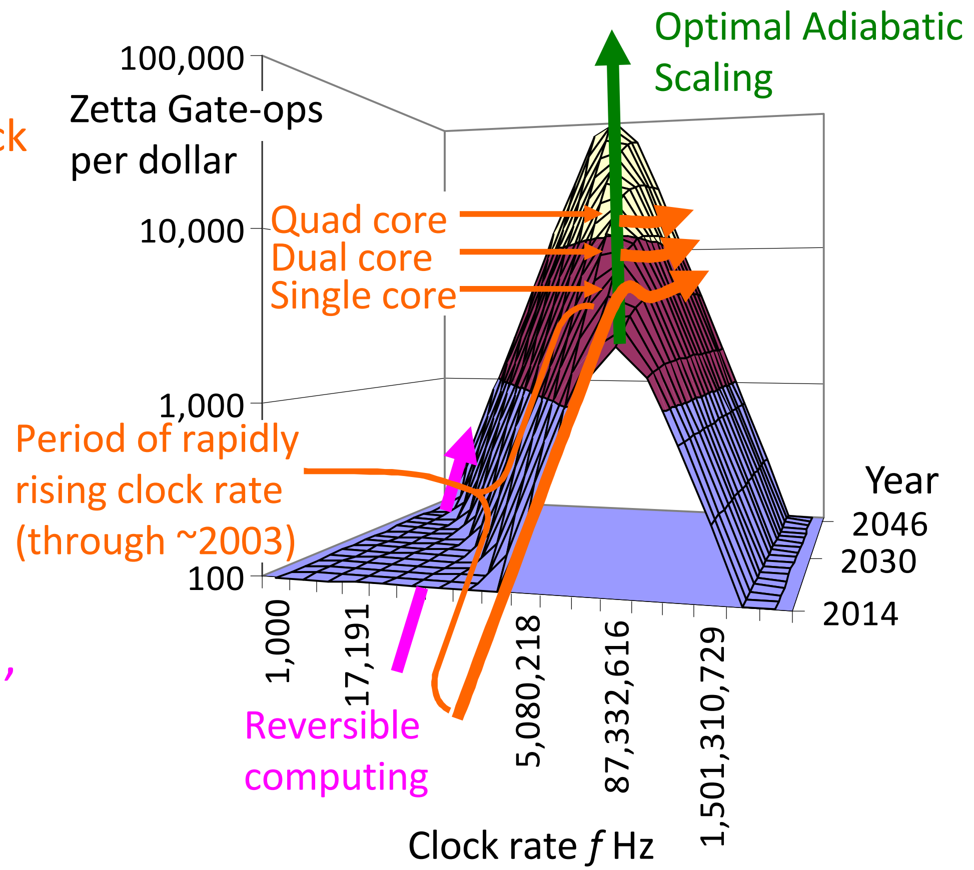
$\$_{energy} = Cf^2$ (*A*, *B*, and *C* constants)

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



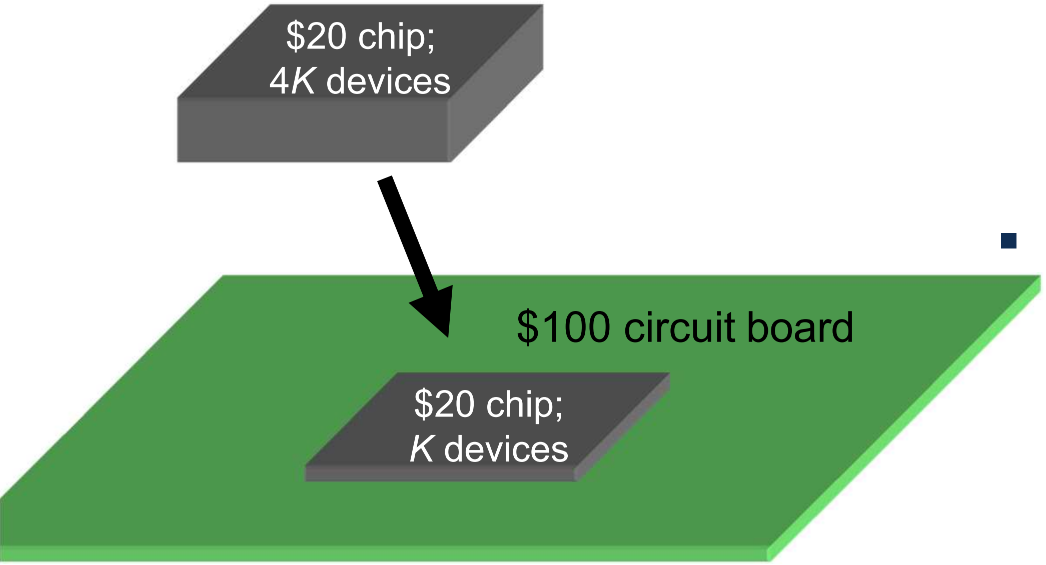
Backup: historical context and reversible computing

- 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/√4× (halve)
  - Power per device drops to 1/4×
  - Power per chip stays same
  - Throughput doubles: 4× as many devices runn at 1/√4× the speed, for a net throughput increase of √4×
- "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.

Under optimal adiabatic scaling, 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	$1/\alpha$	$N=\alpha^2$[†]
V	$1/\alpha$	1	1	$\boxed{1}$	1	$\boxed{1}$
C	$1/\alpha$	$1/\alpha$	$1/\alpha$	$\boxed{1}$	$1/\alpha$	$\boxed{1}$
$U_{stor} = ½\ CV^2$	$1/\alpha^3$	$1/\alpha$	$1/\alpha$	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	$\boxed{1}$	α	$\boxed{\sqrt{N}=\alpha}$

← 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

Sandia National Laboratories

Outline

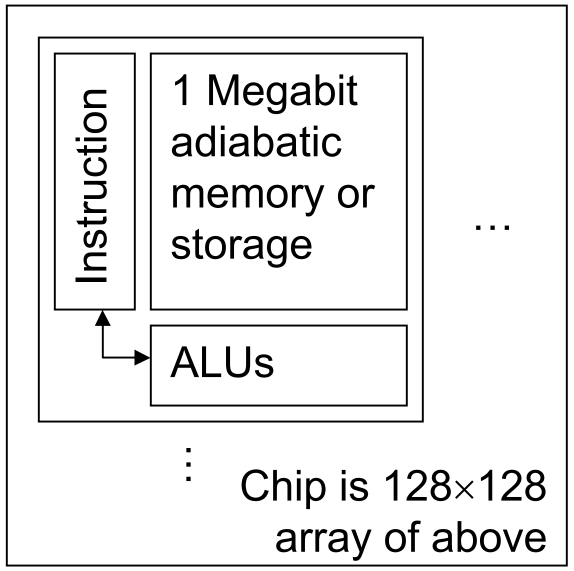
- Preview
- Improving power efficiency without changing devices
- Architecture
- Programming
- Performance analysis of example
- Computer system model with integrated I/O

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

Backup: 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):

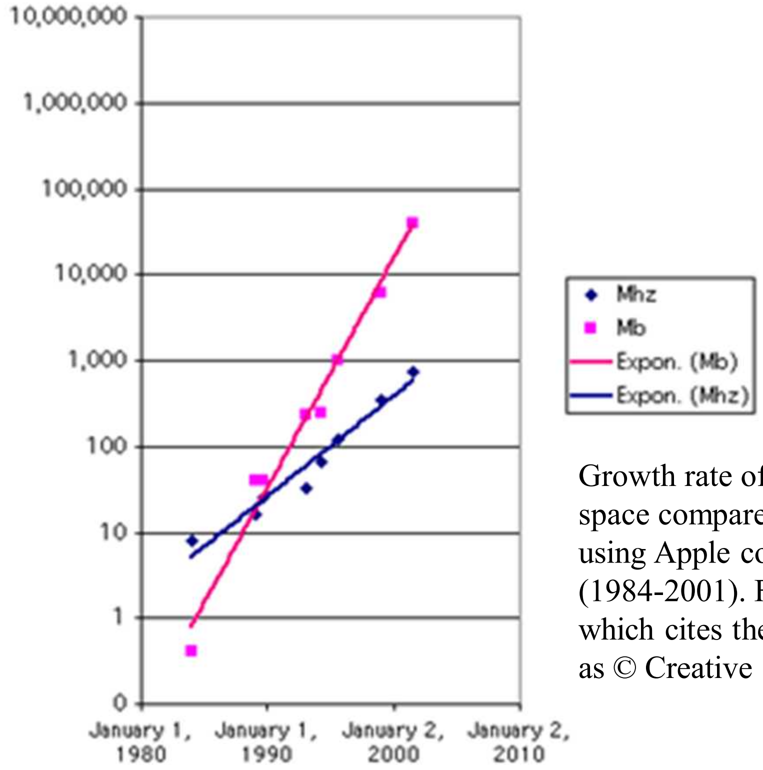

Instruction
1 Megabit adiabatic memory or storage
ALUs
…
⋮
Chip is 128×128 array of above

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--

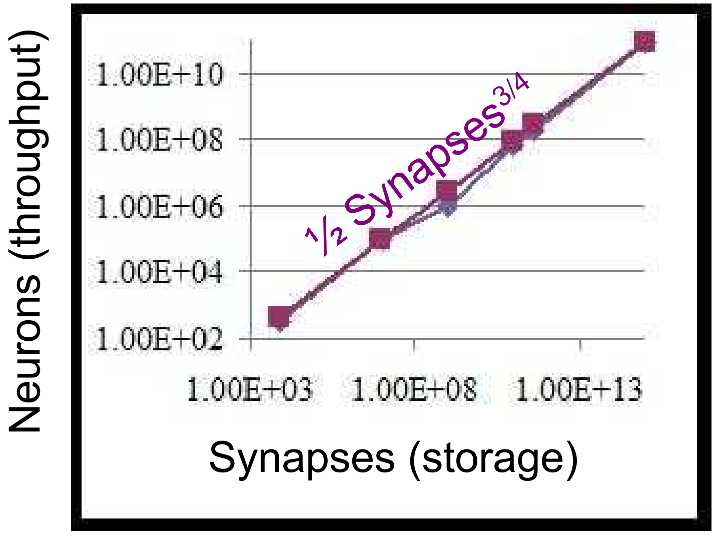
What applications scale like PIMS?

- 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

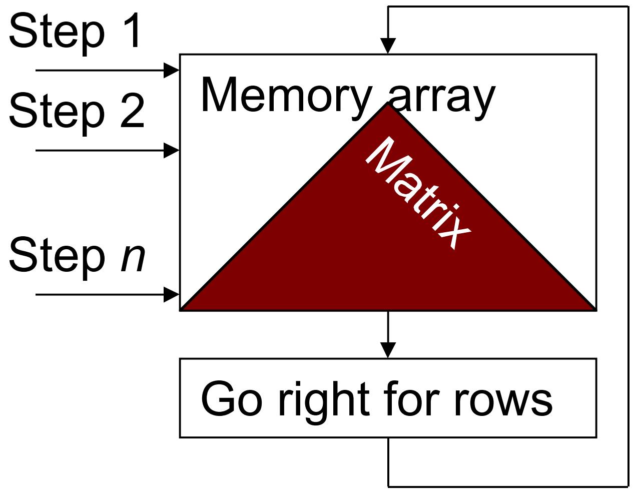
Source: Wikipedia

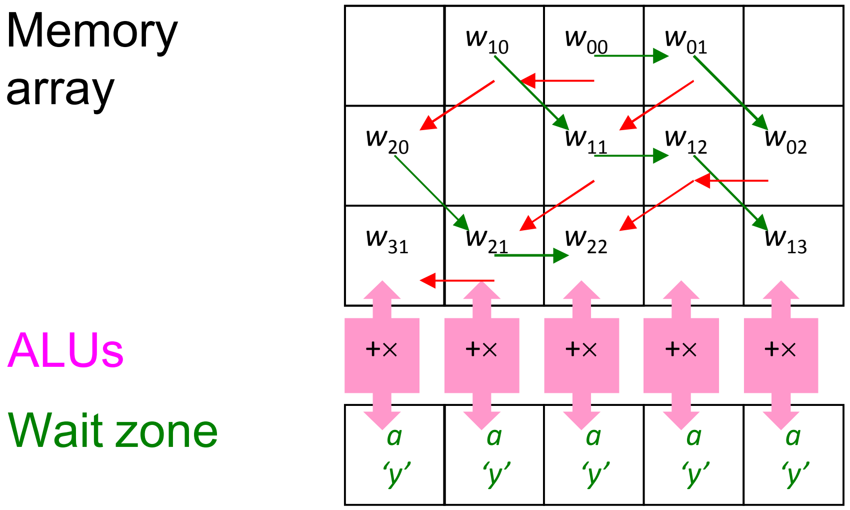
Outline

- Preview
- Improving power efficiency without changing devices
- Architecture
- Programming
- Performance analysis of example
- Computer system model with integrated I/O

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

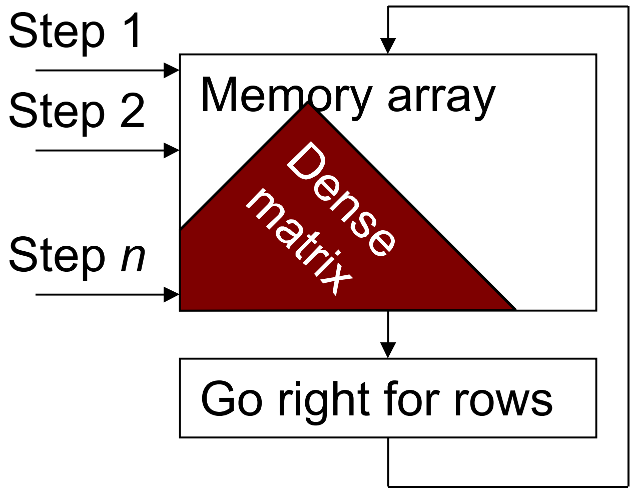




Memory array
ALUs
Wait zone
w_{10} w_{00} w_{01}
w_{20} w_{11} w_{12} w_{02}
w_{31} w_{21} w_{22} w_{13}
+×
a
'y'

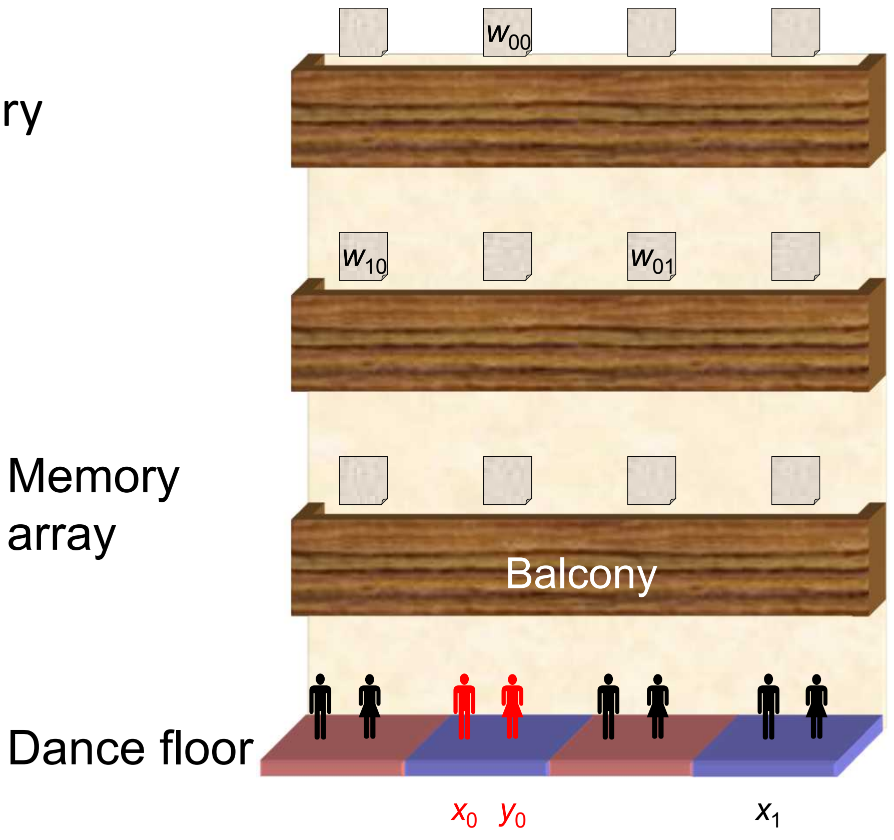
Programming a dense vector-matrix multiply

- Init: Ladies have vector element; gents have zero accumulation
- Program: Ladies multiply memory output by their vector element, pass to gent; gent adds to accumulating sum; ladies step right; gents step left



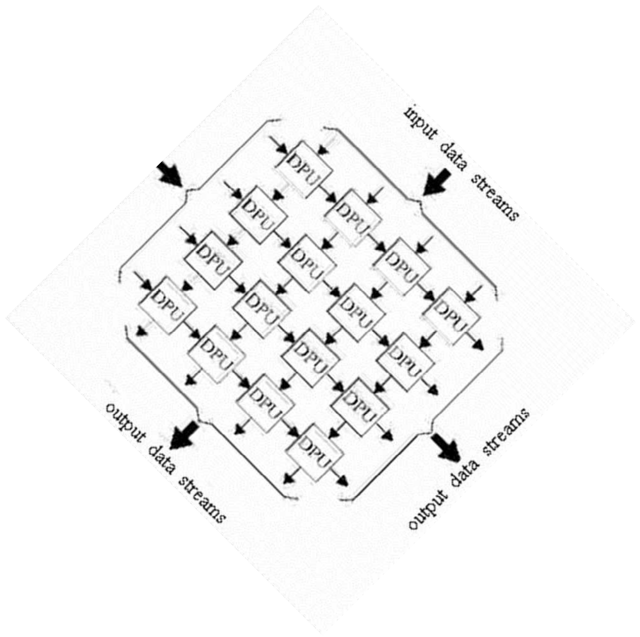
Wx = y; gent w_{00} x_0 then w_{10} x_0; lady $y_0 = w_{00}\ x_0 + w_{01}\ x_1$

- Dance hall model



Note: This program only uses half the memory locations; better algorithm would use a hexagonal layout, but is too complex for PowerPoint

Backup (embedded spreadsheet)



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:			🚶 x_0 1		🚶 y_0 0			
Timestep 2:		🚶 x_1 2		a00 1 🚶 x_0 1 🚶 y_0 1		🚶 y_1 0		
Etc.	🚶 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		
1st cell column above, as it evolves with time	🚶 y_1 12		a32 0 🚶 x_3 4 🚶 y_2 6		a23 5 🚶 x_2 3 🚶 y_3 17			
	2nd cell column above, as it evolves with time	🚶 y_2 6		a33 0 🚶 x_3 4 🚶 y_3 17				
		3rd cell, and so on	🚶 y_3 17					

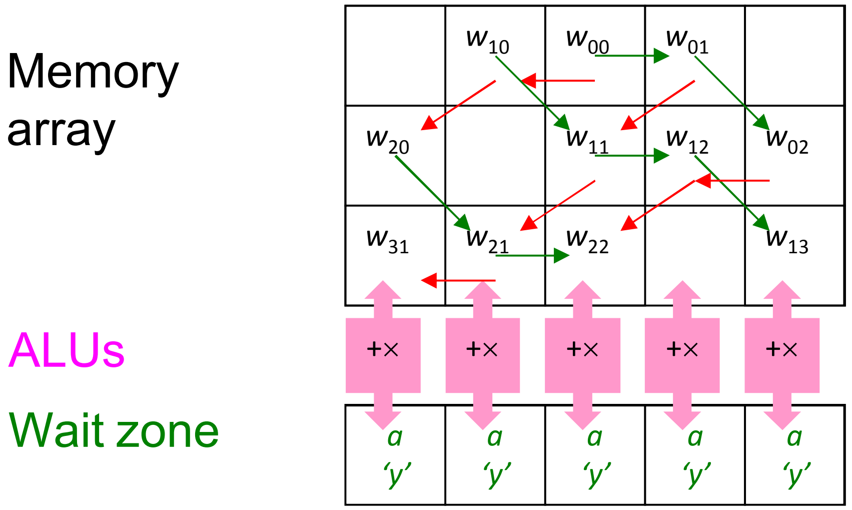
Note: the y_j's are updated, so they do not all have the same value

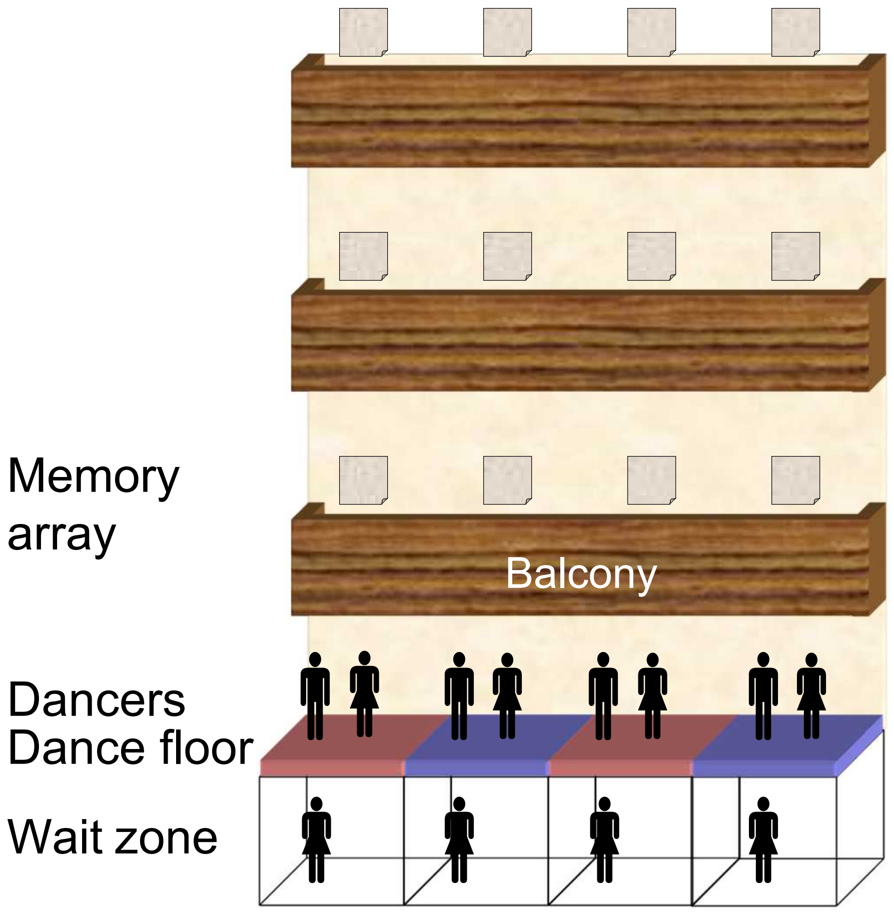
Note on above: this diagram is only a spreadsheet, but you may think of a row of x's and y's as a register that shifts right and left each time step; the a's do not shift (see arrows).

Extreme Multiple Instruction Multiple Data (MIMD)

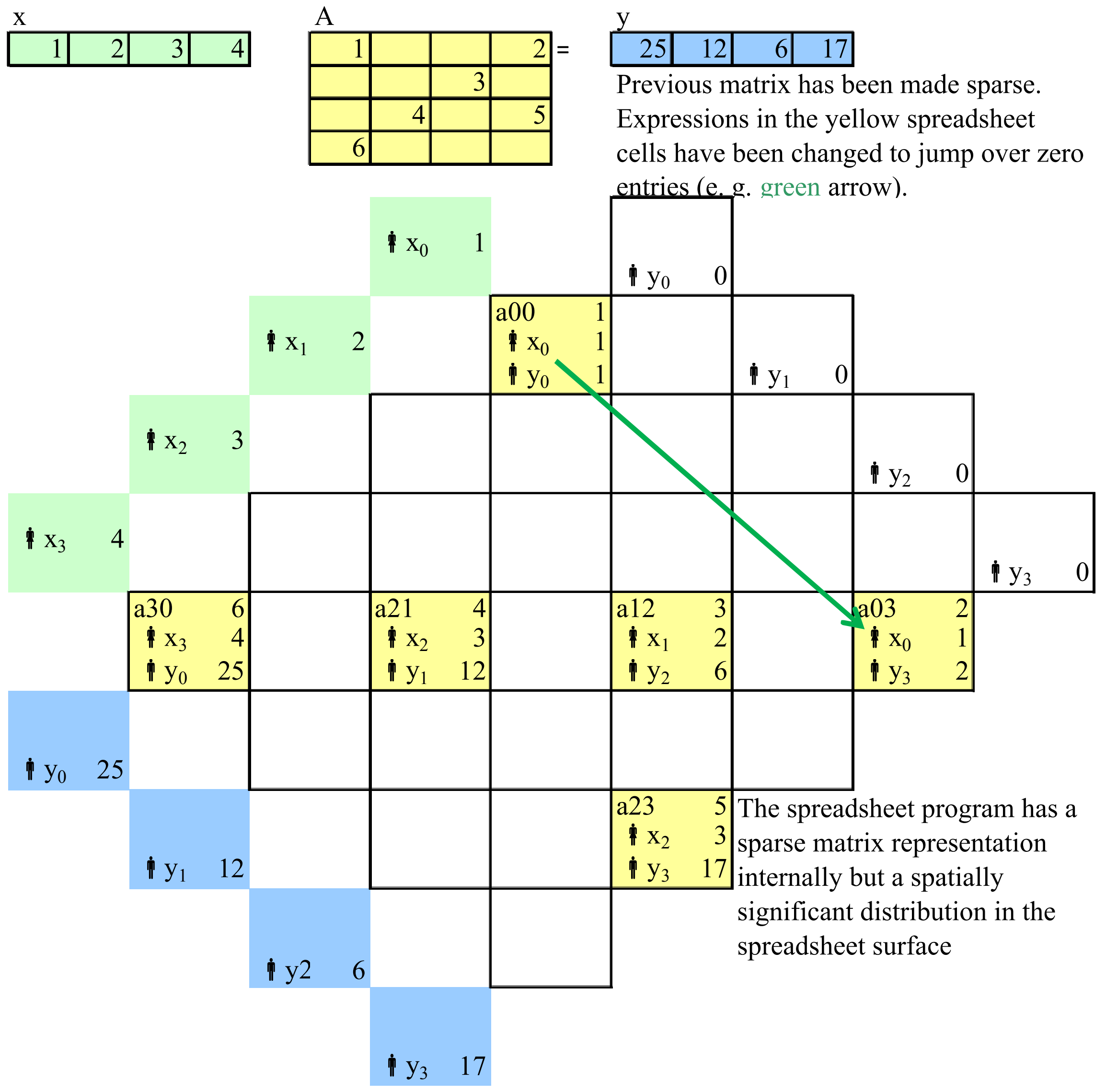
- Ladies and gents are additionally given an “appointment card” telling them to appear n_1 steps away n_2 steps later
- The appointment card may require them to wait in a wait zone

- Dance hall model


Memory array
w_{10} w_{00} w_{01}
w_{20} w_{11} w_{12} w_{02}
w_{31} w_{21} w_{22} w_{13}
ALUs
+× +× +× +× +×
Wait zone
a ‘y’ a ‘y’ a ‘y’ a ‘y’ a ‘y’



Backup (embedded spreadsheet)



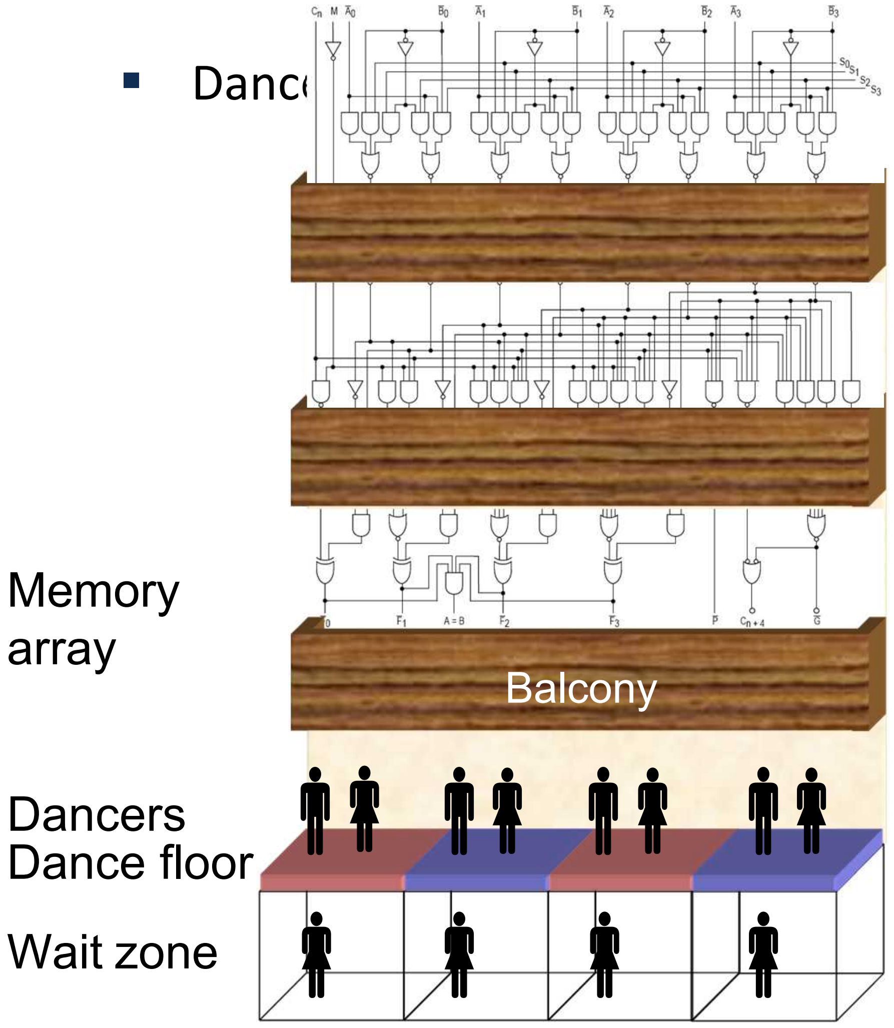
Previous matrix has been made sparse. Expressions in the yellow spreadsheet cells have been changed to jump over zero entries (e. g. green arrow).

The spreadsheet program has a sparse matrix representation internally but a spatially significant distribution in the spreadsheet surface

General programming

- The memory array holds a representation of a general function, using operators and communications.
- Ladies, gents, dogs, cats, etc. implement the primitive operators and interconnect
- Possible types of operators
  - Booleans logic gates (FPGA)
  - Arithmetic (register transfer)
  - Perceptrons (neural network)

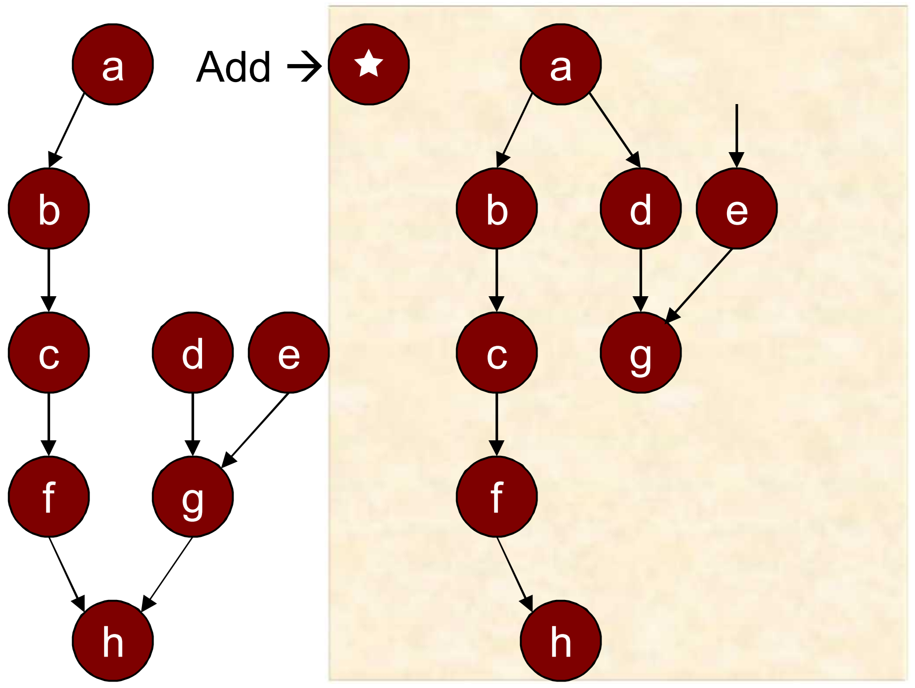
- Danc



Compiling

- System has graph layout functions built in
  - Sugiyama (GraphViz)-type algorithms
  - With dynamic addition/ removal
- Activities illustrated
  - Layout a parse tree
  - Add a new operation to an existing parse tree (animation)

- Dance hall model



Parse tree

Control codes in PIMS memory

Backup (embedded spreadsheet)

x

1	2	3	4

A

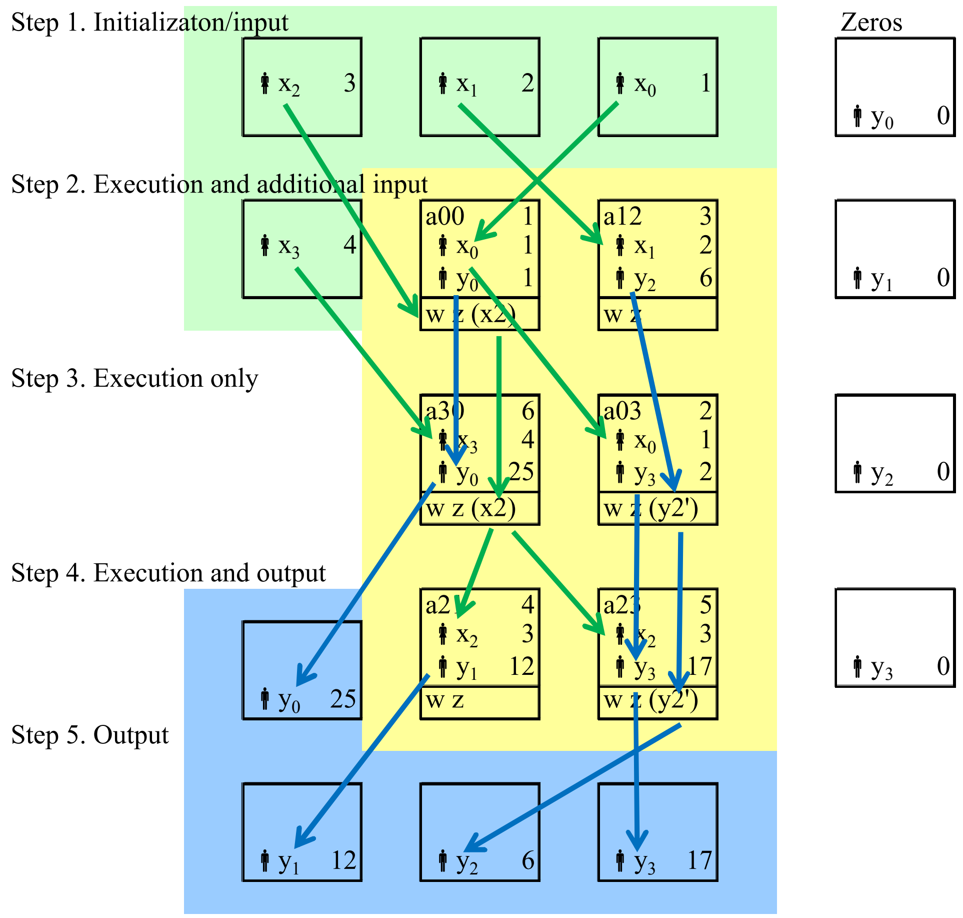
1			2
		3	
	4		5
6			

=

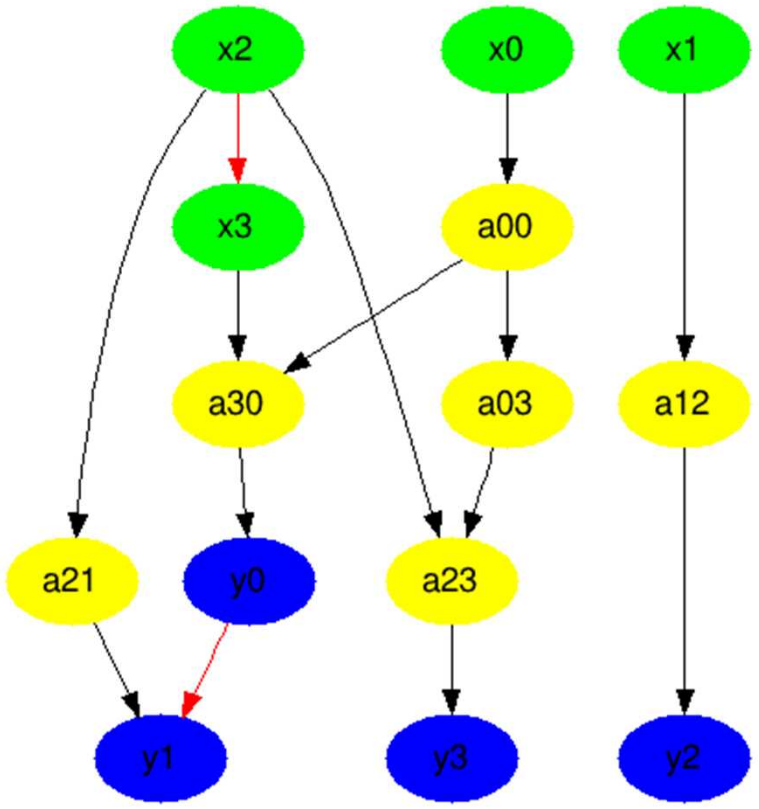
y

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:



Programming model

- Like a self-modifying FPGA
  - You start with the equivalent of an empty FPGA
  - To program it, you feed it a list of gates to be added incrementally to the gate diagram inside
    - May also delete gates
  - The PIMS chip itself optimizes the placement in real time
  - When done programming, you can run the PIMS like an FPGA
- Like a neural network
  - You don't just store collections of data to be perused by a von Neumann machine somewhere else
  - Instead, you store data in the form of its use, such as storing a table of data along with instructions on how to search it

Outline

- Preview
- Improving power efficiency without changing devices
- Architecture
- Programming
- Performance analysis of example
- Computer system model with integrated I/O

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

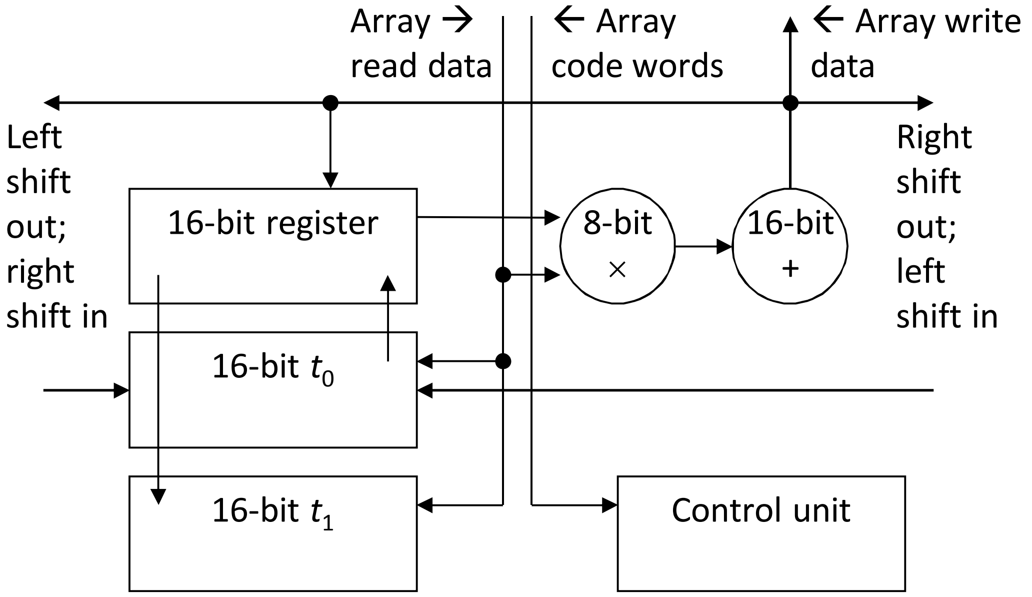
Exemplary ALU

- 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

Memory Logic type	GTX 750 Ti 0.1 nj/bit	DRAM 46.0 fj/bit	Adiabatic Mem 0.9 fj/bit
TFET 1.3 fj/synapse 12 bits needed	1.0 nj 0.0 j 1.0 nj 20.8 mw	552.0 fj 1.3 fj 553.3 fj 11.1 kw	10.9 fj 1.3 fj 12.2 fj 244.3 w
CMOS HP 21.8 fj/synapse 12 bits needed	1.0 nj 0.0 j 1.0 nj 20.8 mw	552.0 fj 21.8 fj 573.7 fj 11.5 kw	10.9 fj 21.8 fj 32.7 fj 653.2 w
TFET 21 bits 7.7 fj/synapse 25 bits needed	2.2 nj 0.0 j 2.2 nj 43.4 mw	1150.0 fj 7.7 fj 1157.6 fj 23.2 kw	22.7 fj 7.7 fj 30.4 fj 607.9 w
CMOS HP 21 bits 127.8 fj/synapse 25 bits needed	2.2 nj 0.0 j 2.2 nj 43.4 mw	1150.0 fj 127.8 fj 1277.7 fj 25.6 kw	22.7 fj 127.8 fj 150.5 fj 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)

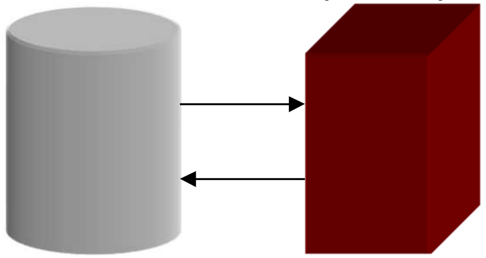
Note: NVIDIA GTX 750 Ti is memory bandwidth limited so the logic energy is ignored.

Outline

- Preview
- Improving power efficiency without changing devices
- Architecture
- Programming
- Performance analysis of example
- Computer system model with integrated I/O

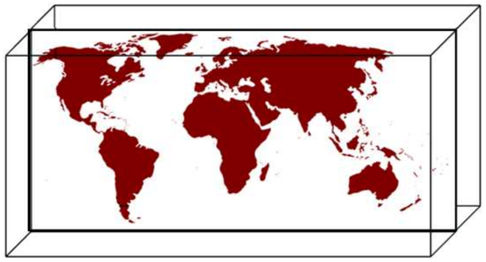
Data model for Processor-In-Memory-and-Storage (PIMS)

A. von Neumann model with input/output:



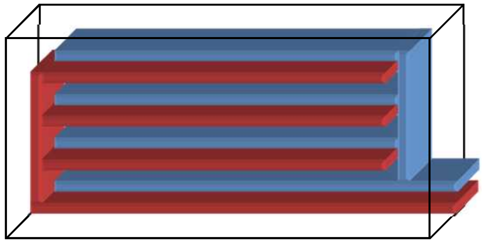
Read input
Parse
Process with √N efficiency boost
Format
Write output

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~~

Is this a memory technology or a processor technology?

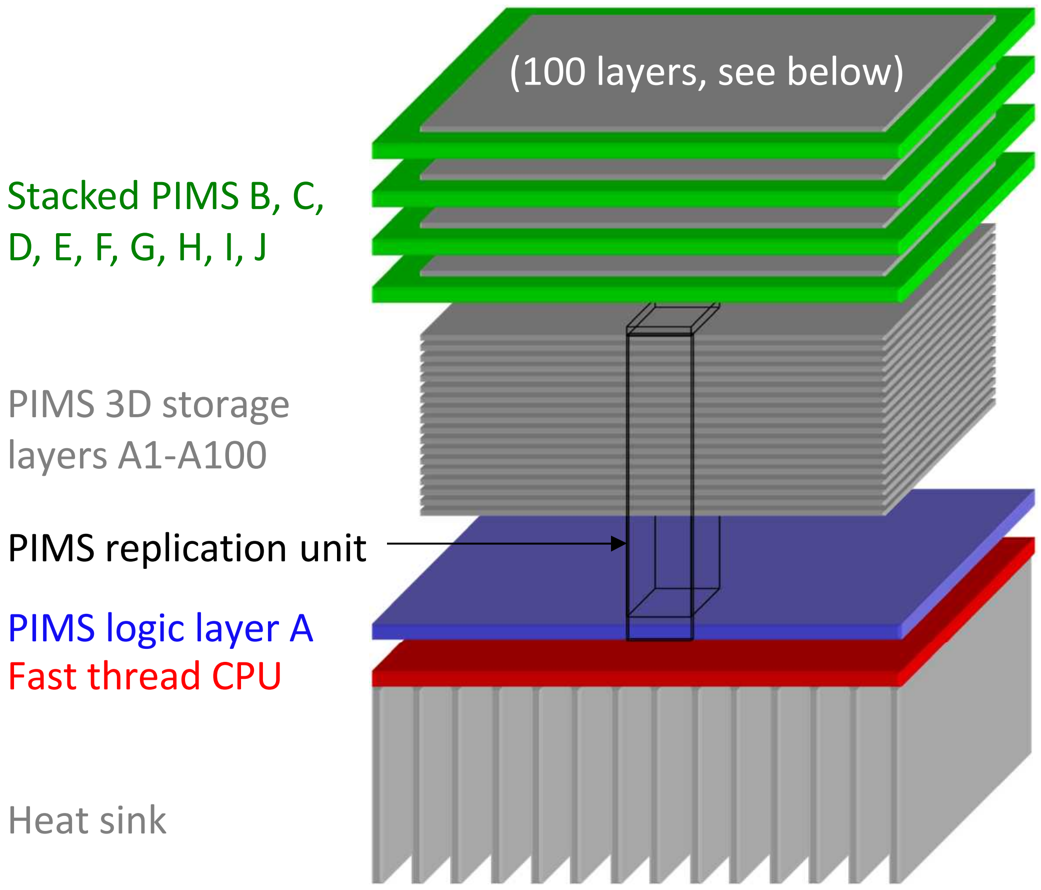
Answer: Both

- PIMS + optimal adiabatic scaling applies to processing node and memory
  - If problem AND DATA have parallelism, PIMS + optimal adiabatic scaling can exploit it with full power-efficiency boost discussed
  - If problem, data, or algorithm lack parallelism, the available throughput boost shifts from $\sqrt{N}$ to 1 uniformly
    - Actually $N^{\delta/2}$, where data dimensionality is δ
    - A fully serial program has δ=0
- Brains get away without a fast thread accelerator, but it became an impediment so we invented the computer
- So I propose a system with a spectrum of speeds

Final summary

	Fast CPU	Gen 1	Gen N
Clock	3 GHz	100 MHz	10 MHz
Devices	10^{10}	10^{13}	10^{15}
Stack × Layers	1 × 1	10 × 100	Molecular assembly?
Ops/joule	1×	30×	300×
Fast thread penalty	.1		
Parallelism boost		3000	30,000
Total throughput	1×	30,000×	300,000×
Power	100W	100W	100W

Exploded view:



Conclusions

- Is "Moore's Law ending"?
  - Continued manufacturing cost reductions by exploiting 3D have a lot of upside
  - Whether to call it Moore's Law is a marketing decision
- 3D and new device
  - A new transistor-like device is unlikely to restart Moore's Law (not in talk)
  - However, 3D manufacture could restart Moore's Law even with CMOS
  - New devices could be useful for other reasons
    - Devices for other functions, like memory
    - New transistor-like devices whose benefit is more efficient manufacture
- Programming
  - Presented one programming example in this talk (deep neural network)
  - One example meets programmability standard of parallel computers at introduction
  - Question: Is a deep learning neural network Turing complete? Hmmm. Alan Turing used his deep learning neural network to create the Turing Machine as a tool, forming an argument that a neural network is as general as a Turing Machine