Energy Efficiency of Computing Architectures

A Deep Dive into Processors and Emerging Computing Machines

Olivier Sentieys
Inria
Univ. Rennes, Irisa
olivier.sentieys@inria.fr

http://people.rennes.inria.fr/Olivier.Sentieys

Cairn Team at a Glance

Inria Rennes

Enssat Lannion

- *Energy-Efficient Computing Architectures*

- ~35 people, Rennes and Lannion campuses
- INRIA, Univ. Rennes 1, ENS Rennes
- Electrical Engineering & Computer Science

- Domain-specific computing architectures
- Design tools and compilers
- Wireless, signal, image, security

Energy Efficiency Challenges

- Teraops/Watt?
  - 10^{12} op./s/W ≡ 1 pJ/op
  - Several orders of magnitude from current processors and multicores
- From Sensors 🌡 to Clouds ☁
  - 1 TOPS @ 1W
    - Clouds, embedded systems
  - 1 GOPS @ 1mW
    - IoT sensor nodes
  - 1 MOPS @ 100μW
    - Energy harvesting

Intel CPU+FPGA

Micro Mote

Improving Energy Efficiency

- Technology?
  - What can advanced technology nodes bring
- Accelerate 🌡 ☁
  - Energy advantages of specialized hardware
- Approximate 🌡 ☁
  - Playing with accuracy to reduce energy
- Manage the Power 🌡 ☁
  - Dynamic Voltage/frequency (Over-)Scaling
  - Energy Harvesting sensor nodes

Key Questions

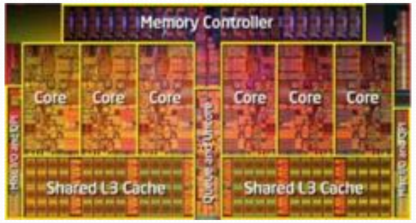
- A deep dive into processors… *(I hope not too deep)*
- Basics on transistors, logic gates, registers, memory
- Energy consumption of processor core/uncore
- Computers are parallel
  - Billions of transistors doing the job at the same time
  - Are multicore processors the solution?
- Specializing the computer
  - Reconfigurable computing
- Emerging paradigms
  - Neuromorphic, approximate, stochastic

Outline

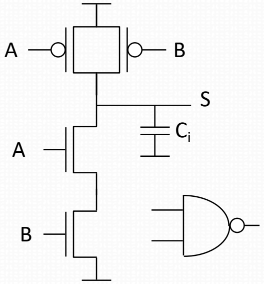
- Part I: From Transistors to Logic Gates
  - Basic Element, Delay, Power Consumption
  - The Issue of Synchronization
- Part II: Inside a Processor
  - Von-Neumann Architecture, Instruction Set Architecture, Operating Systems
  - Multicore Processors, Power and Utilization Walls
- Part III: Pushing the Accelerator!
  - Hardware Accelerators
  - Reconfigurable Computing
- Part IV: Emerging Computing Paradigms
  - Neuromorphic Computing
  - Approximate Computing
  - Chips are going 3D

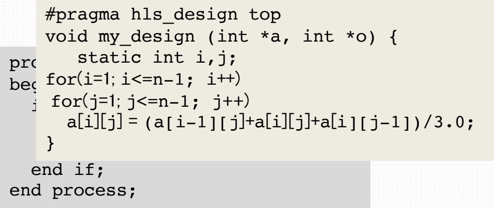
Integrated Circuit Design

- Chips, logic gates and transistors



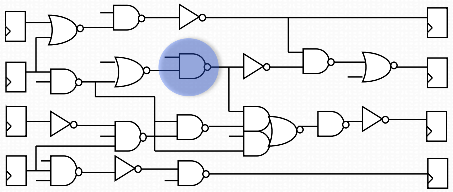
Intel's Xeon Chip





```
#pragma hls_design top
void my_design (int *a, int *o) {
    static int i,j;
for(i=1; i<=n-1; i++)
 for(j=1; j<=n-1; j++)
   a[i][j] = (a[i-1][j]+a[i][j]+a[i][j-1])/3.0;
}
```

```
pro
beg
  i
  end if;
end process;
```

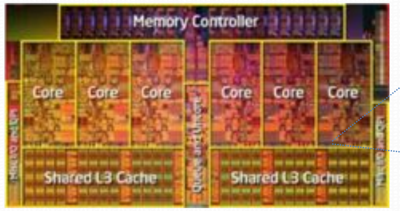


Part I: From Transistors to Logic Gates

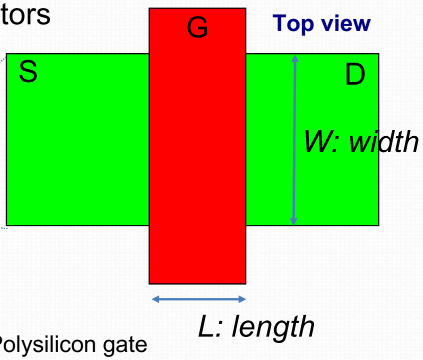
- The Fundamental Element: MOSFET Transistor
- Design of CMOS Cells: Combinatorial Logic
- Memory Cells
- Delay and Power Consumption
- Synchronous Design

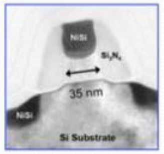
Fundamental Building Block: MOSFET Transistor

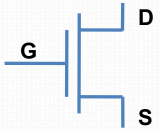
Now several billions or transistors

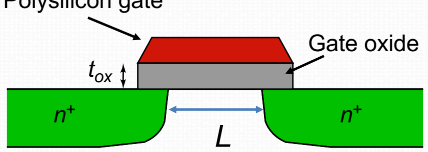


Intel’s Xeon Chip







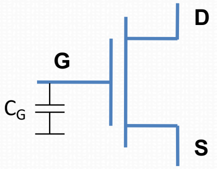


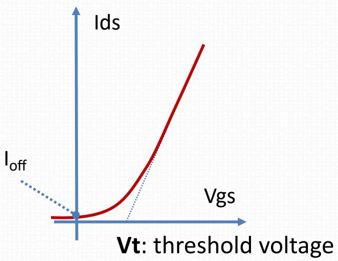
Cross section

MOSFET: Metal Oxide Silicium Field Effect

The Basic Element: Transistor

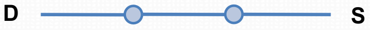
- Transistor as a switch



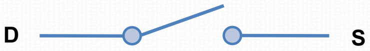


Vt: threshold voltage

- Vgs > Vt: NMOS on
  - Resistance R_{DS}



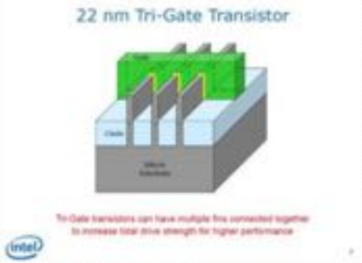
- Vgs < Vt: NMOS off
  - Leakage I_{off}

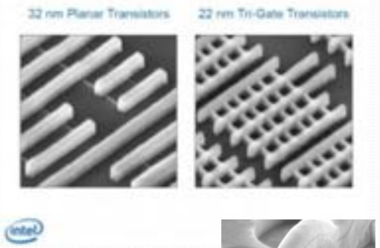


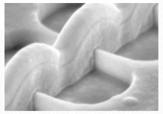
- Gate: capacitance C_G
- Switch: resistance R_{DS}

Transistors Nowadays

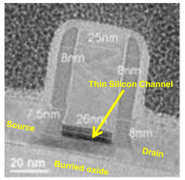
- Intel FinFET: transistors go 3D


22 nm Tri-Gate Transistor
Tri-Gate transistors can have multiple fins connected together to increase total drive strength for higher performance
intel





- Fully Depleted SOI[1]
  - Low-power



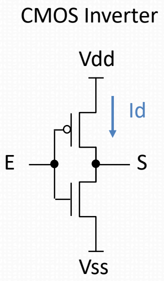
[1] Silicon on Insulator

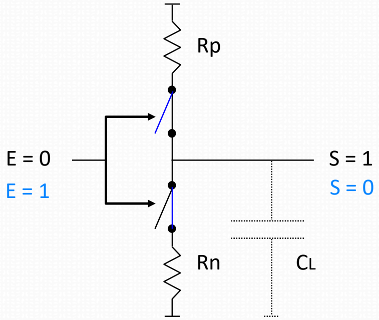
Part I: From Transistors to Logic Gates

- The Fundamental Element: MOSFET Transistor
- Design of CMOS Cells: Combinatorial Logic
- Memory Cells
- Delay and Power Consumption
- Synchronous Design

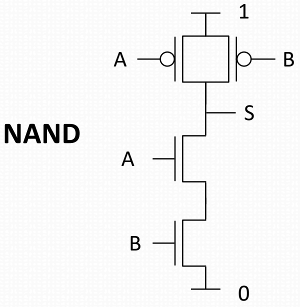
Combinatorial Logic Cells

- Complementary Logic (CMOS)

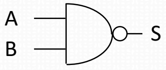


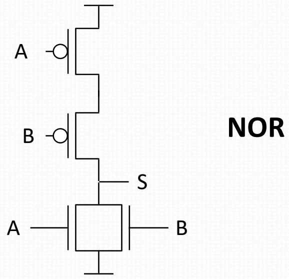


NAND and NOR

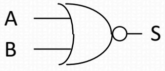


A	B	S
0	0	1
0	1	1
1	0	1
1	1	0



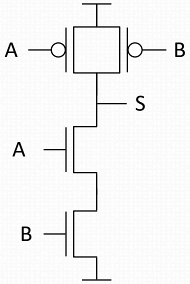


A	B	S
0	0	1
0	1	0
1	0	0
1	1	0

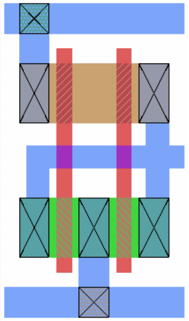


Layout Design

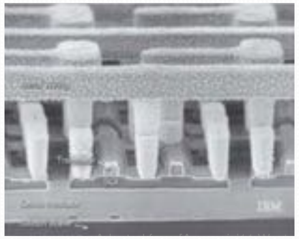
Transistor Schematic



Layout



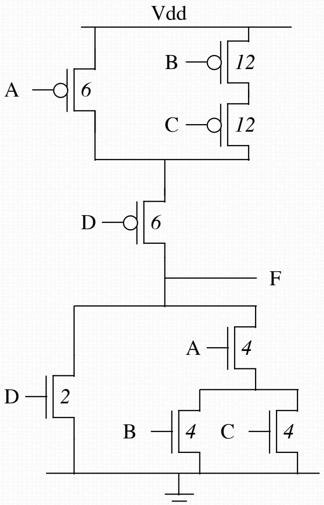
Silicon



Complex Gates

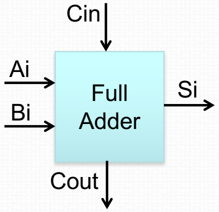
- $F = \overline{A.(B+C)+D}$

- The art of transistor sizing
  - Equilibrate delay for $0 \rightarrow 1$ and $1 \rightarrow 0$ output transitions
  - Minimize cell area

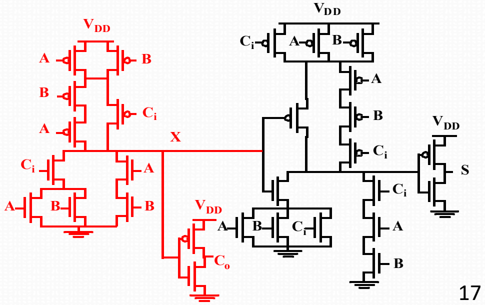


Complex Gates: Full Adder

- Full Adder

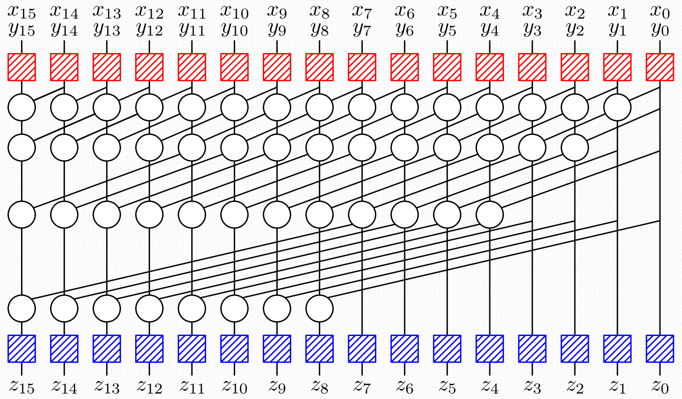


Ai	Bi	Ci	Co	Si
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



Complex Functions

- 16-bit Adder (integer)



Part I: From Transistors to Logic Gates

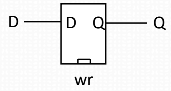
- The Fundamental Element: MOSFET Transistor
- Design of CMOS Cells: Combinatorial Logic
- Memory Cells
- Delay and Power Consumption
- Synchronous Design

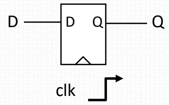
Storing Values

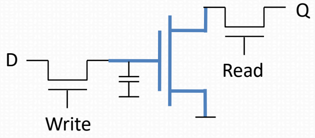
Capacitor (DRAM)

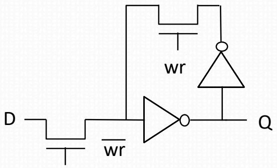
Latch (SRAM)

Flip-Flop (Register)

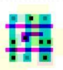


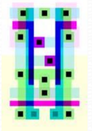


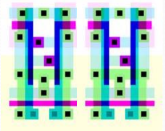




- Setup Time: Tsetup
- Hold Time: Thold
- Propagation Time: Tp

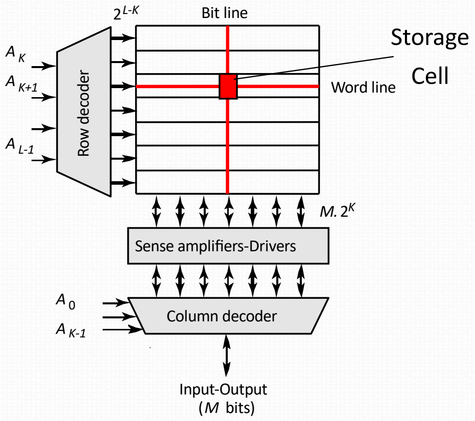


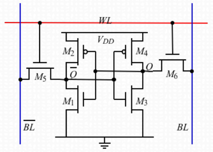


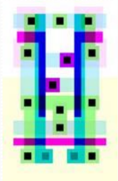


Memory

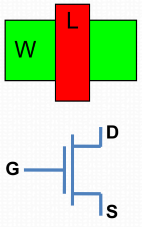
- L2 Cache contains 4 Millions SRAM cells
  - Raw/column of 2000 cells

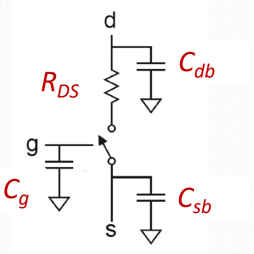


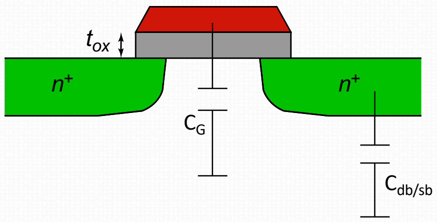




Delay: Parasitic Elements







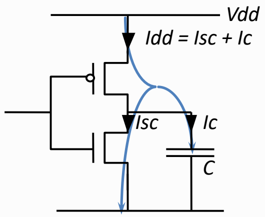
- Drain-Source Resistance: $R_{DS} = \frac{L}{W}\frac{1}{k(V_{dd}-V_t)}$
- Gate Capacitance: $C_g = \frac{\epsilon W.L}{t_{ox}} = W.L.C_{ox}$

$$\text{Delay} \propto R_{DS}.C_g \propto \frac{L^2}{Vdd-Vt}$$

Power and Energy Consumption

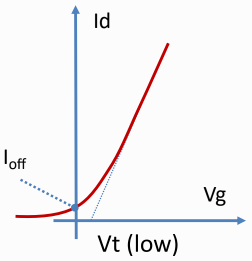
- Dynamic power
  - Charge and discharge of node capacitance
- Energy = $\mathbf{C.Vdd^2}$
- Power

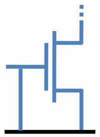
$$\mathbf{P_{dyn_i} = C.Vdd^2.f.Prob_{0\rightarrow 1}}$$



- Static power: *Ps*
  - Sub-threshold and junction leakage current

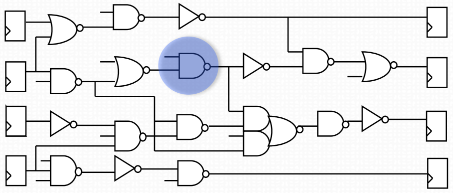
$$\mathbf{P_{stat_i} = N.I_{off}.Vdd}$$





Power at Higher Level

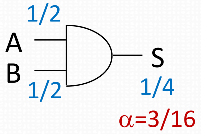
- Propagating activity



$$\mathbf{P = \sum_i [\alpha_i.f_i.C_i.Vdd^2 + I_{leak_i}.Vdd]}$$

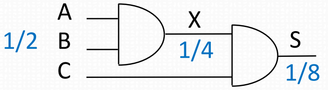
Activity

- Activity α_i is the probability to have a *0→1* transitions at the output of a gate
- Example: AND gate
  - $P_S = P(S=1) = P_A P_B$
  - $\alpha_i = P_S(1 - P_S)$



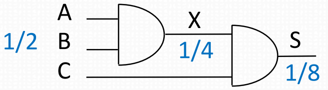
A	B	S
0	0	0
0	1	0
1	0	0
1	1	1

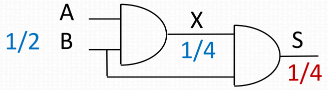
- Activity propagation



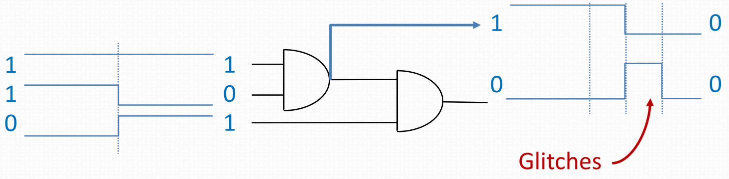
Propagating Activity is not So Simple

- Conditional probabilities





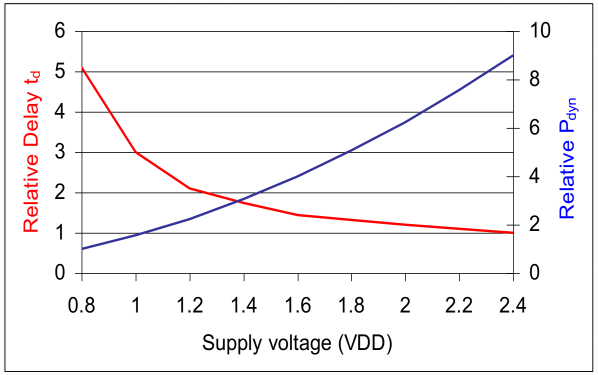
- Glitches: gate delay
  - Significant in arithmetic



Dynamic Power vs. Performance

- Decreasing Vdd reduces power but increases delay

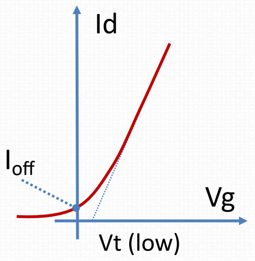
$$\mathbf{P_{dyn_i} = \alpha_i . f_{clk} . C_i . Vdd^2}$$



$$\mathbf{Delay \propto \frac{1}{V_{dd} - V_t}}$$

Leakage vs. performance

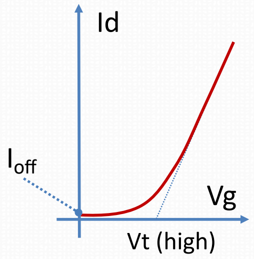
- High performance



$$\mathbf{P_{stat_i} = N . I_{off} . Vdd}$$

$$\mathbf{Delay \propto \frac{1}{V_{dd} - V_t}}$$

- Low leakage

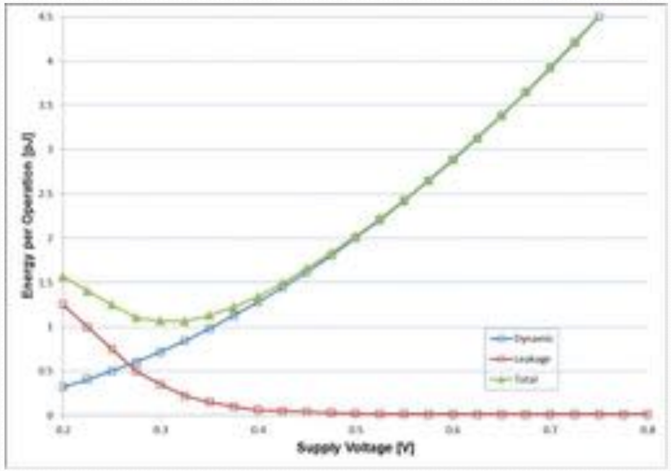


Ioff:

- Exponential in inverse of Vt
- Exponential in temperature
- Linear in device count

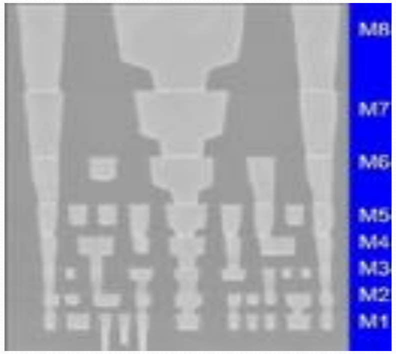
Minimum Energy per Operation

- Putting all together



On-Chip Interconnect?

- Gate delay decreases but… wire delay increases
- Crossing chip in 5-10 clock cycles
- Also affected by noise…



- Metal layers to reduce wire delay
- Repeaters

- Towards network-on-chip

Conclusion: Power in CMOS

$$P = \sum_{i} \left[\alpha_i.f_i.C_i.Vdd^2 + I_{leak_i}.Vdd\right]$$

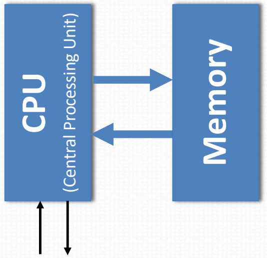
- Dynamic power
  - 40-70% today
  - Decreasing relatively
  - DVFS becomes more and more difficult
- Leakage power
  - 20-50 % today
  - Increasing rapidly
    - number of transistors
    - Vdd/Vt scaling
  - Critical for memory

$$\boxed{P = \frac{energy}{operation} \times rate + static\ power}$$

Inside (Simple) Processor Architecture

Von Neumann Computers

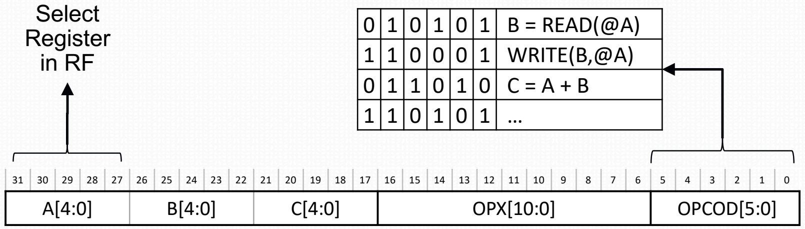
- Processing address, data, control, on the same resources
- Single memory for data and program
- Sequential behavior



- Practically, most processors use Harvard model: separated data and program memory

Instruction Set Architecture (ISA)

- ISA defines a programmer’s interface
- Each instruction is defined by coding (binary) and semantics

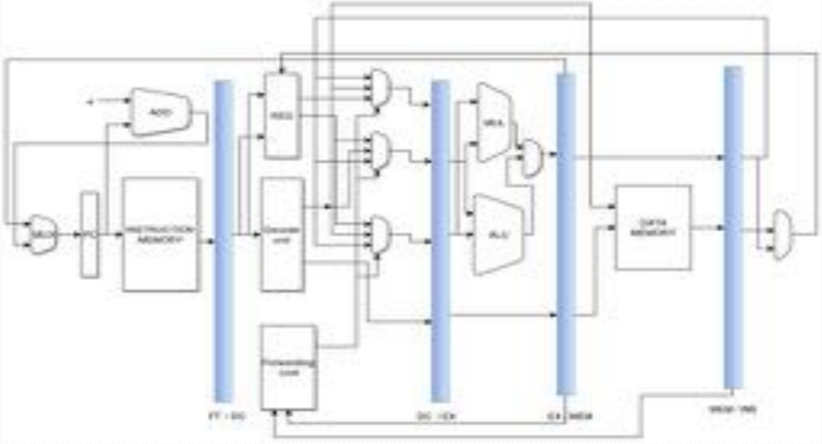


0	1	0	1	0	1	B = READ(@A)
1	1	0	0	0	1	WRITE(B,@A)
0	1	1	0	1	0	C = A + B
1	1	0	1	0	1	…

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A[4:0]					B[4:0]					C[4:0]					OPX[10:0]											OPCOD[5:0]					

Microarchitecture Pipeline

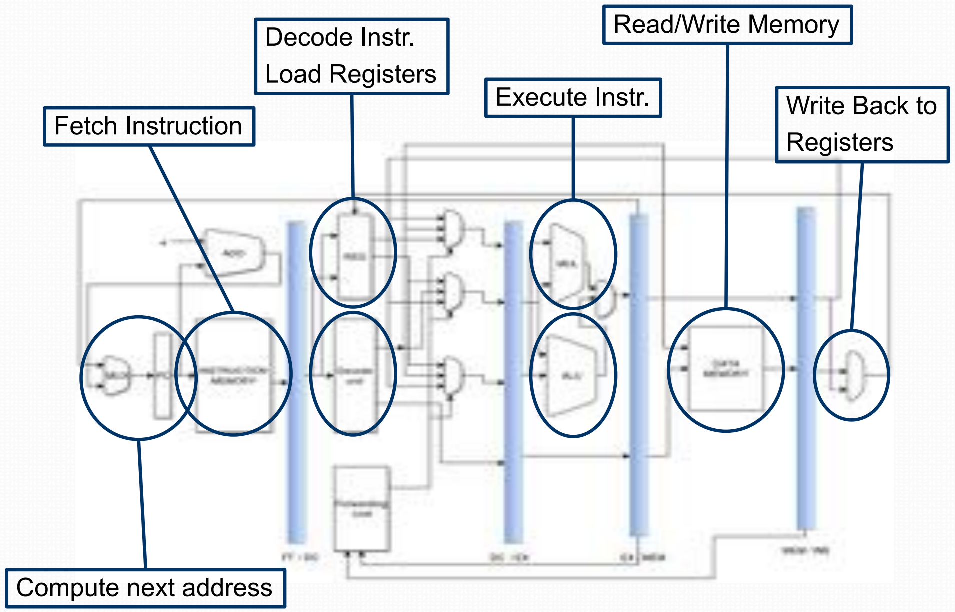
- Microarchitecture defines how instructions are executed (not unique)



Execution of an instruction involves

1. Instruction fetch
2. Decode and register fetch
3. ALU operation
4. Memory operation (optional)
5. Write back (optional)

and compute address of next instruction



Decode Instr.
Load Registers

Read/Write Memory

Execute Instr.

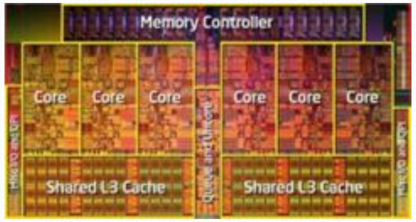
Write Back to
Registers

Fetch Instruction

Compute next address

Achieving Higher Performance

- Branch/value prediction
- Cache memory
- In-core parallelism
  - Multiple Fus
  - Out of order execution
  - VLIW+good compilers



- Multiple cores on a single chip

Abstraction in Computer Systems

- Maximum of an array T

```
numpy.amax(T)
```

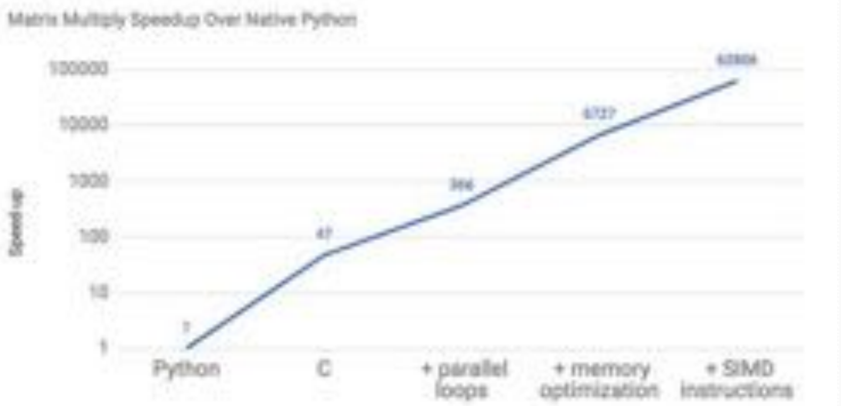
0	1	0	1	0	0	1	0	1	0	1
1	1	0	0	0	1	1	0	0	0	1
0	1	1	0	1	0	1	1	0	1	0
1	1	0	1	0	1	1	0	1	0	1

```
int largest(int T[], int length) {
  int max = T[0];
  for(i=1; i<length; i++) {
      if (max < T[i]) {
            max = T[i];
      }
  }
  return max;
}
```

```
        R1 ← *R2        // max
loop    R2 ← R2+1       // T[]
        R3 ← *R2
        R1 < R3 ?
        BZ next
        R1 ← R3
next    B loop
```

Abstraction and Performance?

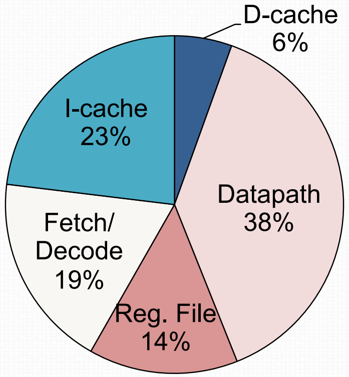
- Matrix Multiply: relative speedup to a Python version (18 core Intel)



["There's Plenty of Room at the Top," Leiserson, et. al.]

Energy Cost in a Processor

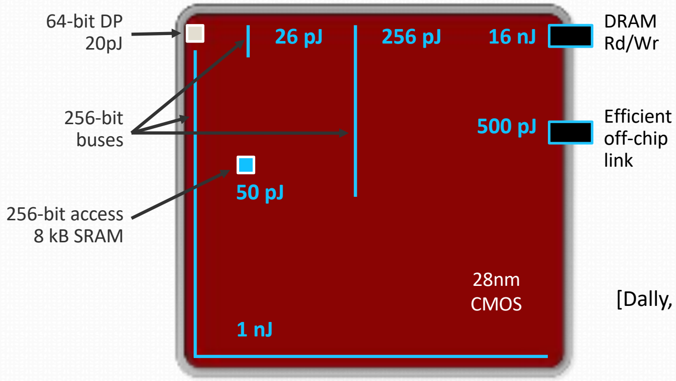
- Operation:
  - 32-bit addition: 0.05pJ
  - 16-bit multiply: 0.25pJ
  - 64-bit FPU: 20pJ/op
- Instruction:
  - fetch, decode, read 2 operands from RF, execute, write back



MIPS Proc.
91 pJ/instr.

Energy Cost in a Processor

- Fetching operands costs more than computing

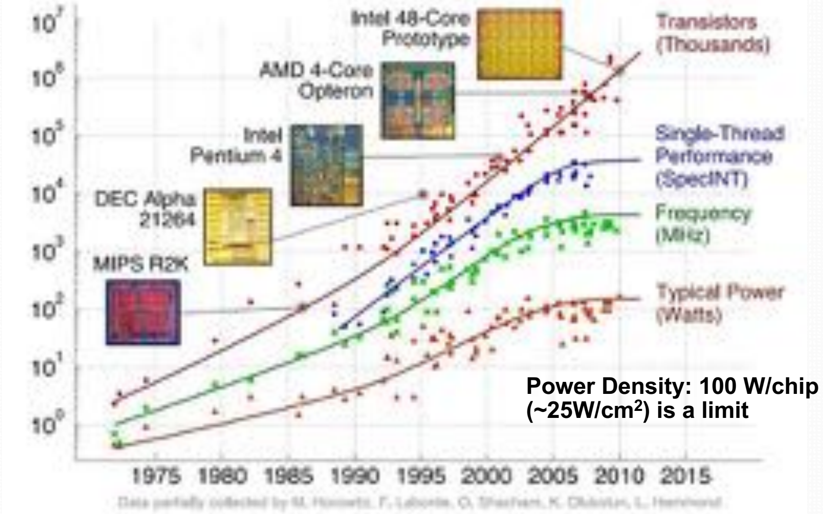


[Dally, IPDPS'11]

Multicore: it's all a trick!

Power and Utilization Walls

And then came the "Power Wall"

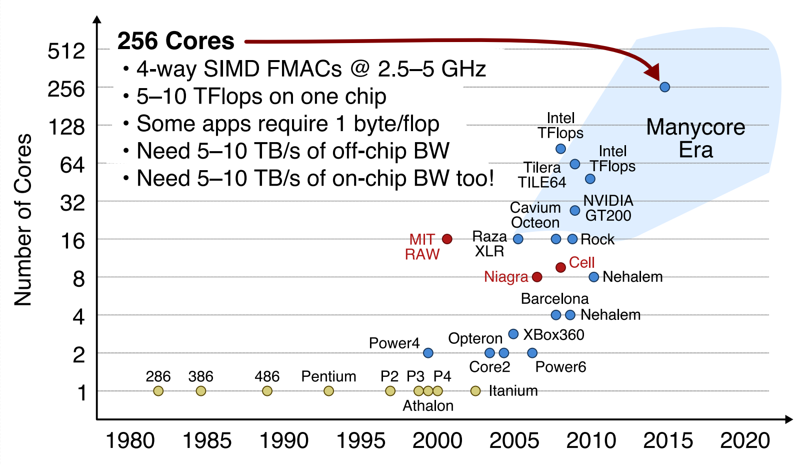


Power Density: 100 W/chip (~25W/cm^2) is a limit

Source: C. Batten, Cornell

and the "Multicore Era"

- Increasing performance by increasing # of cores

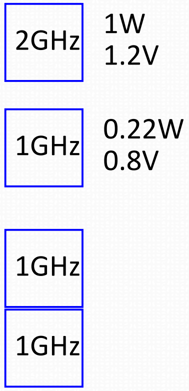


256 Cores
- 4-way SIMD FMACs @ 2.5–5 GHz
- 5–10 TFlops on one chip
- Some apps require 1 byte/flop
- Need 5–10 TB/s of off-chip BW
- Need 5–10 TB/s of on-chip BW too!

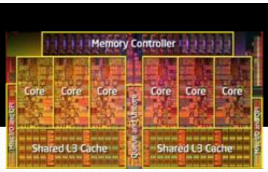
–Source: C. Batten, Cornell

Moving to multicore

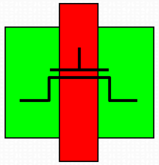
- 1 core@2GHz@1.2V@1W
- 1 core@1GHz@0.8V@0.25W
- 2 cores@1GHz@0.8V@0.5W
- But… twice area (and not so simple)
- Advanced technology nodes?



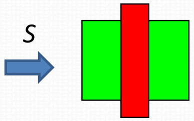
Technology Scaling



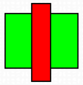
Intel's Xeon Chip



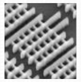
28 nm

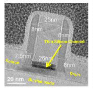


20 nm



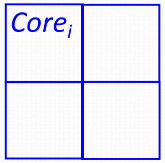
14 nm



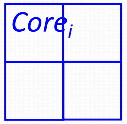


Classical (Dennard's) scaling

Device count	S^2
Device frequency	S
Capacitance, Vdd	1/S
Device power	$1/S^2$
Utilization	**1**



100W@*f*



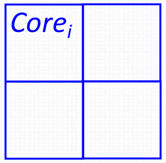
50W@*1.4.f*

End of Dennard's Scaling

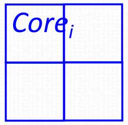
- Energy efficiency is not scaling along with integration capacity

Leakage limited scaling

Device count	S^2
Device frequency	S
Device power (cap)	1/S
Device power (V_{dd})	**~1**
Utilization	**$1/S^2$**

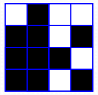


100W@*f*

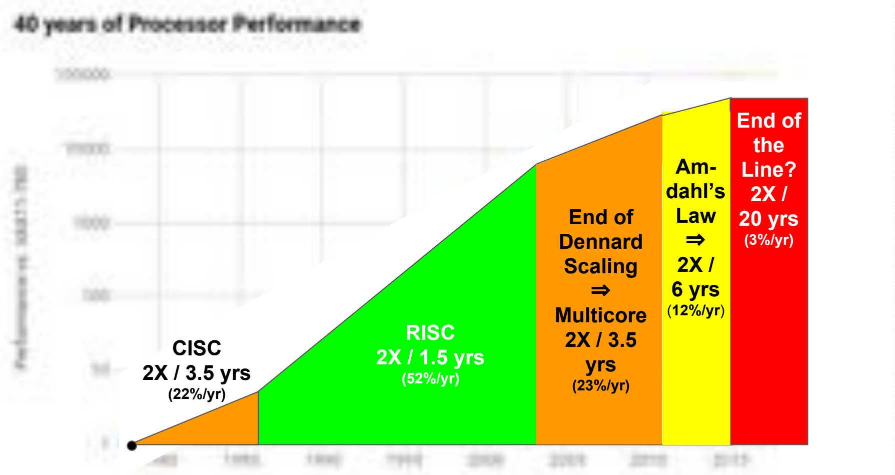


100W@*1.4.f*
(w/o) leakage

- Utilization Wall: percentage of a chip that can switch at full frequency drops exponentially
- Replace dark cores with specialized cores (10-100x more energy efficient)



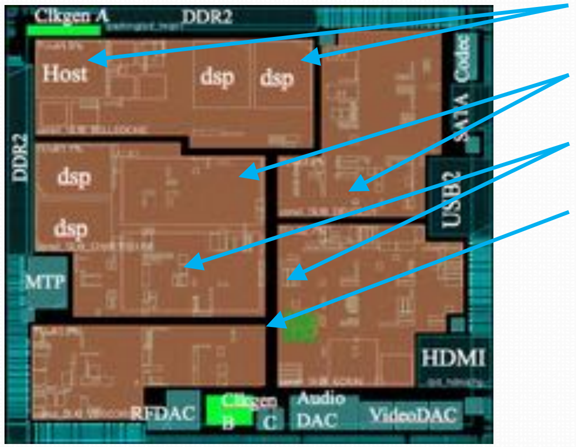
End of Growth of Speed?



Based on SPECintCPU. Source: John Hennessy and David Patterson, Computer Architecture: A Quantitative Approach, 6/e. 2018

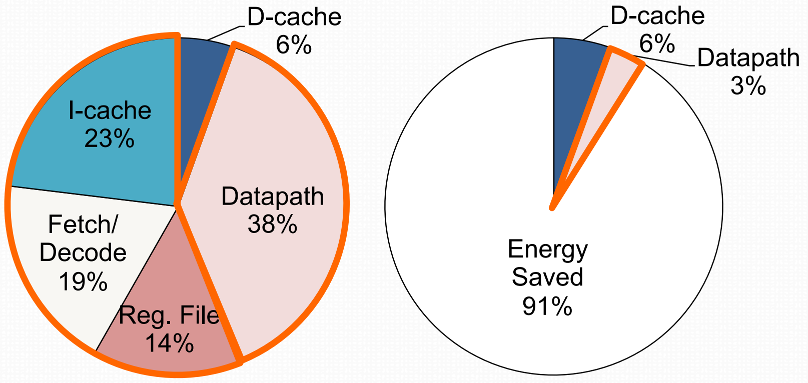
Part III: Pushing the Accelerator!

What is a HW accelerator?



- 16 processors
- 38 HW blocks
- 140 memory blocks
- 5 Gbytes/s on-chip interconnection network

Energy Savings in Specialized HW



MIPS baseline
91 pJ/instr.

Specialized core
8 pJ/instr.

[Goulding et al., Hot Chips'10]

An example: Bitcoin Mining

Type	Model	Mhash/s	Mhash/J	Power (W)
GPP	Intel Xeon X5355 (dual)	22.76	0.09	120
GPP	ARMCortex-A9	0.57	1.14	1.5
GPP	Intel Core i7 3930k	66.6	0.51	130
GPU	AMD 7970x3	2050	2.41	850
GPU	Nvidia GTX460	158	0.66	240
ASIC	AntMiner S1	180.000	**500**	360
ASIC	AntMiner S5	1.155.000	**1957**	590
FPGA	Bitcoin Dominator X5000	100	**14.7**	6.8
FPGA	Butterflylabs Mini Rig	25.200	**20.16**	1250

BITCOIN MINER

Making ANN Inference more Efficient

- Main motivation: AlphaGo consumes around 250,000 Watts!
- Bring Logic and Memory closer
- Compute less precisely

Tensor Processing Unit

ASIC for TensorFlow

Designed by Google

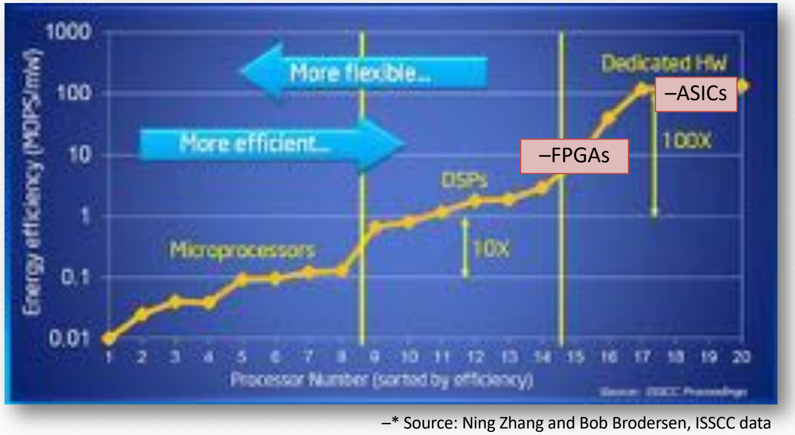
10x better perf / watt

latency and efficiency

bit quantization

- Google Tensor Processing Units (TPU)
  - Computations close to memory
  - 8 bit operations

The Efficiency of Specialization

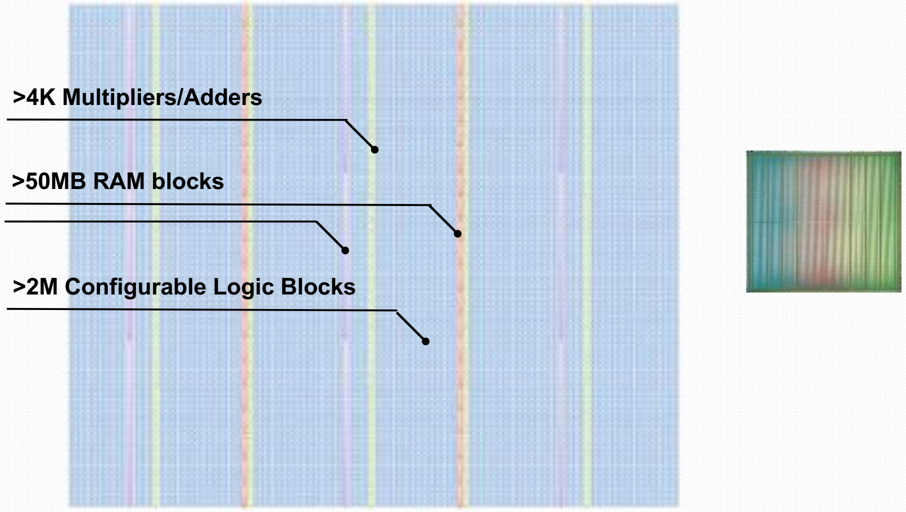


–* Source: Ning Zhang and Bob Brodersen, ISSCC data

–100-1000X Gap in Efficiency … but Specialization comes with Penalties in Programmability

Reconfigurable Hardware Accelerators

Field Programmable Gate Array (FPGA)

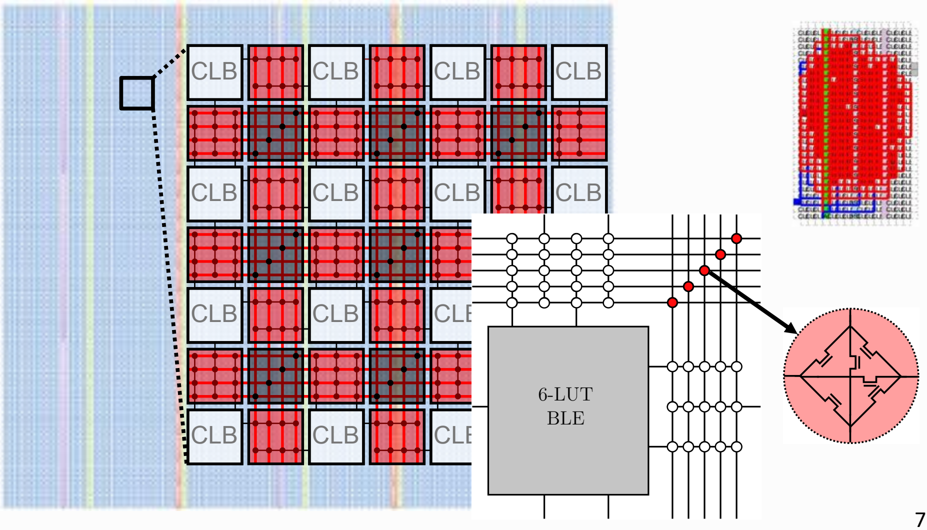


>4K Multipliers/Adders

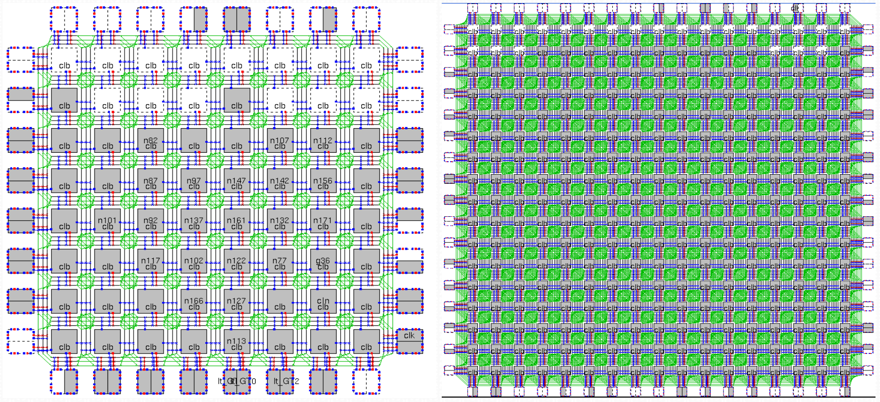
>50MB RAM blocks

>2M Configurable Logic Blocks

Field Programmable Gate Array (FPGA)



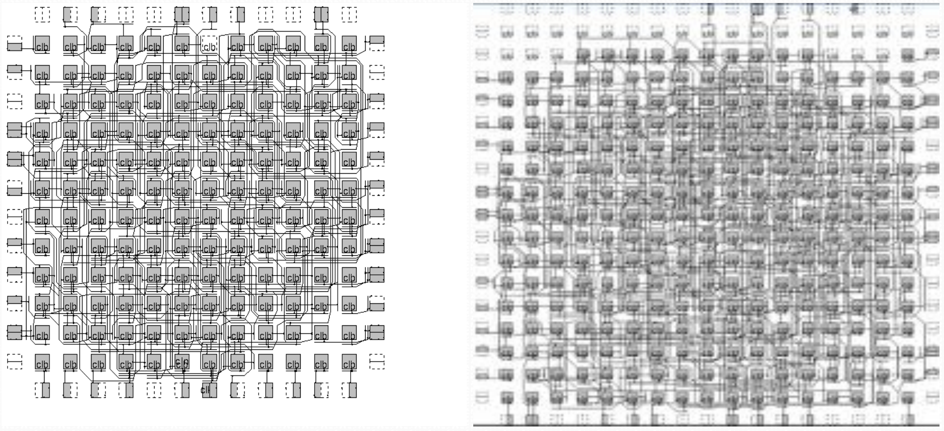
The Program is the Configuration



(a) abs

(b) calcNeighbor

The Program is the Configuration



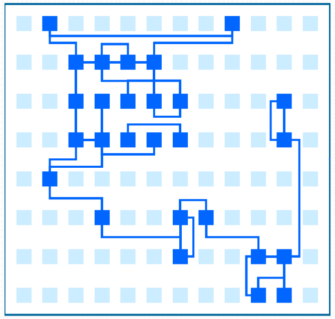
(a) Crc16

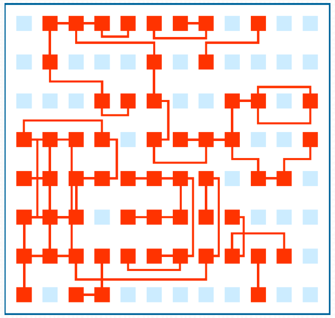
(b) calcNeighbor

Space-Time Computation

```
for(i=1; i<length; i++) {
    if (max < T[i]) {
       max = T[i];
    }
}
```

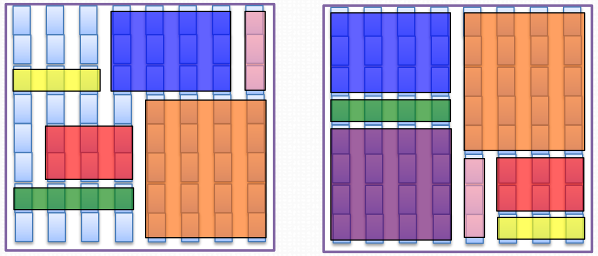
```
for(i=1; i<N; i++) {
   for(j=1; j<M; j++) {
      y[i][j]+=x[i][j]*h[j][i]
   }
}
```





FPGA Acceleration

- FPGAs can run multiple tasks in parallel

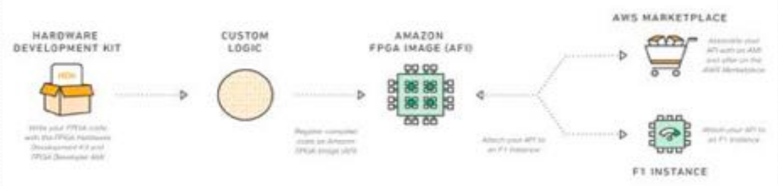


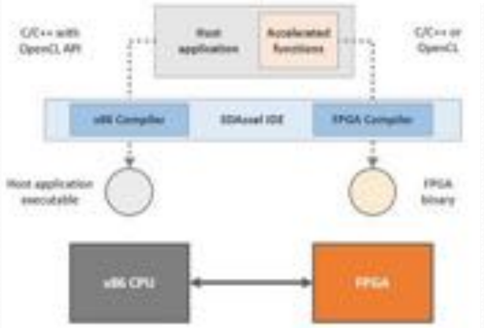
FPGA accelerators
for HPC/Cloud

- Towards heterogeneous multicores

Amazon AWS EC2 F1

aws
aws.amazon.com





AWS EC2 F1 Platform

Instance Size	FPGAs	DDR-4 (GiB)	vCPUs	Instance Memory (GiB)	NVMe Instance Storage (GB)	Network Bandwidth
f1.2xlarge	1	4 x 16	8	122	1 x 470	Up to 10 Gbps
f1.16xlarge	8	32 x 16	64	976	4 x 940	25 Gbps

- Up to 8 Xilinx UltraScale+ FPGA devices in a single EC2/F1 instance

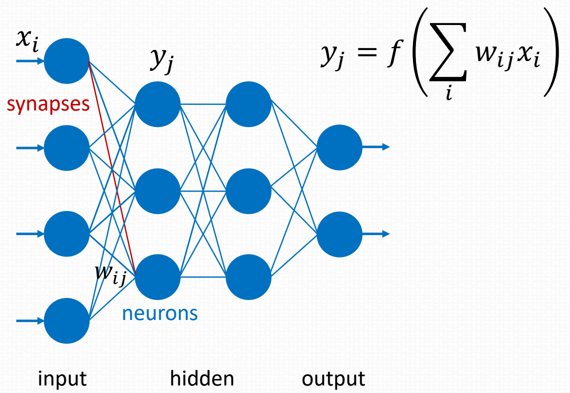
Time has Come for Specialization

- Microsoft Unveils Catapult to Accelerate Bing
  - One FPGA per blade
  - 6 × 8 2-D torus topology
  - High-end Stratix V FPGAs
- Running Bing Kernels for feature extraction and machine learning

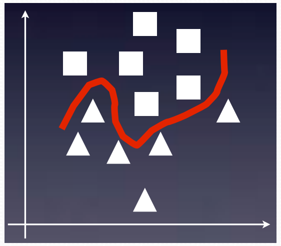
- Increase ranking throughput by 95% at comparable latency to software-only
- Increase power consumption by 10%
- Increase total cost of ownership by less than 30%

Part IV: Emerging Computing Paradigms

How Do Artificial Neural Networks Work?



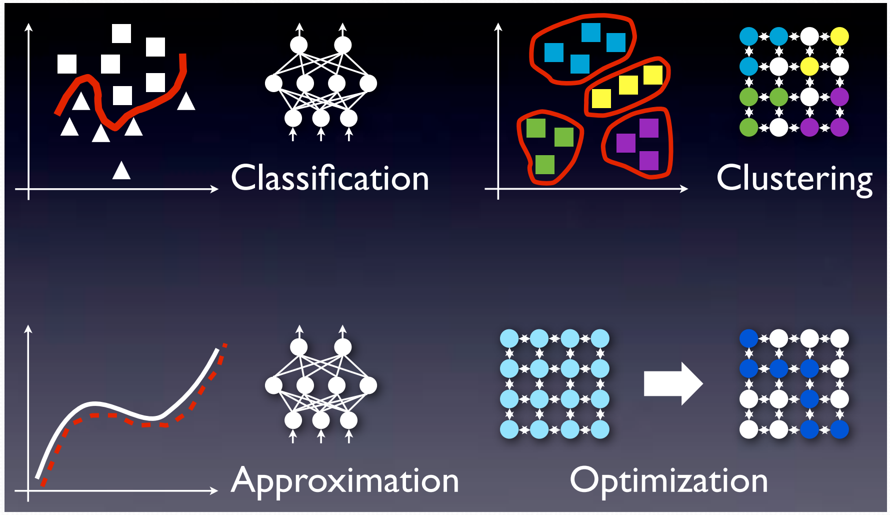
$$y_j = f\left(\sum_i w_{ij} x_i\right)$$



Classifier

- Neural networks are not fundamentally complicated
- The issue: finding the good weights with *learning*

What ANNs Can Do



[O. Temam, ISCA10]

So What's New?

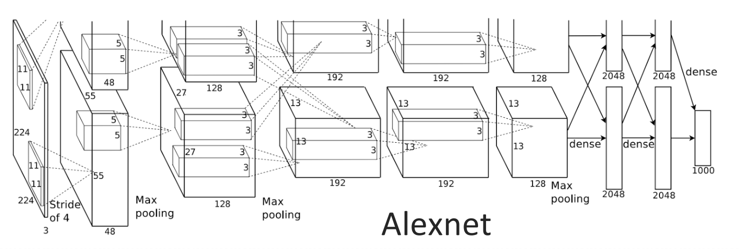
Convergence of trends

- Computer performance (e.g. GPU) can train neural networks with millions of weights
- Access to gigantic datasets
  - Billions of images
  - Training can take weeks!
- More complex ANNs
  - Deep Convolutional Neural Networks (CNN)
  - Long Short-Term Memory (LSTM) Recurrent Neural Networks
- Trendy vision applications
- Emerging technologies offer opportunities

Imagenet

So What's New?

- Deeper Networks

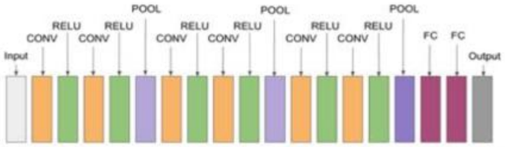


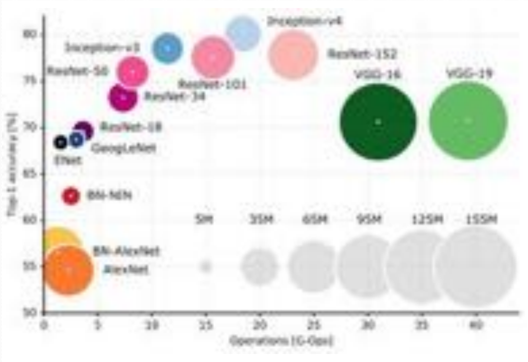
Alexnet

Alex Krizhevsky *et al.*, Imagenet classification with deep convolutional neural networks, 2012.

Complexity of Deep CNNs

- 10-30 GOPS
  - Mainly convolutions
- 10-200 MB
  - Fully-connected layers



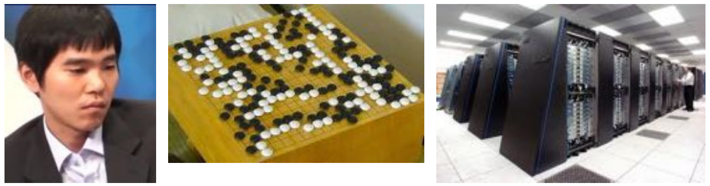


And What About Energy?

- The brain seems to have something very special about energy efficiency

Lee Sedol (brain)

AlphaGo (CPU+GPU with tree seach and deep neural networks)



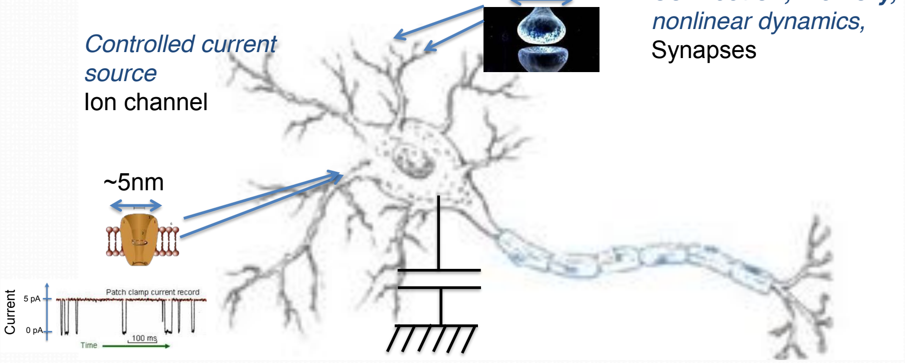
20 Watt

>250 000 Watt

- Computers: arithmetic but chiefly memory transfers

[D. Querlioz, CNRS]

Real Biological Neurons



~500nm

Controlled current source
Ion channel

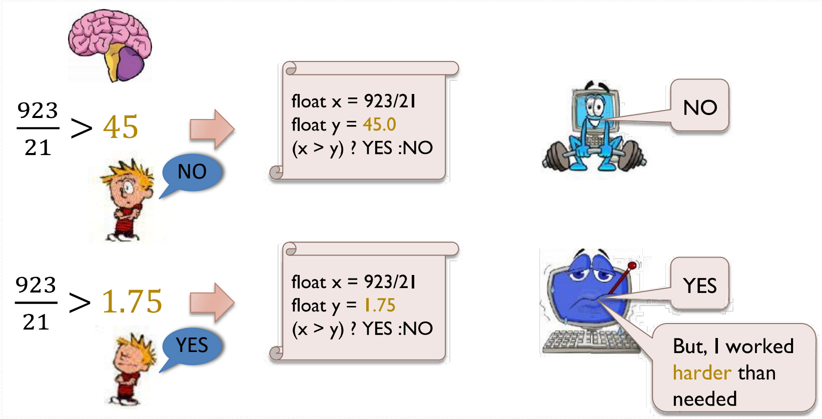
~5nm

Connection, memory, *nonlinear dynamics,*
Synapses

- Brain computes with strong approximations (mostly analog) based on low power, slow, noisy and variable nano-devices

[D. Querlioz, CNRS]

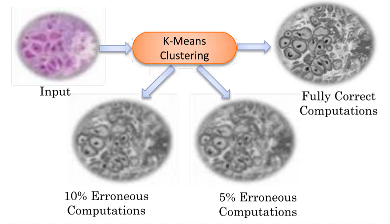
Humans Approximate.... But Computers Do Not!



- Leads to inefficiency
- Overkill (for many applications)

Many Applications are Error Resilient

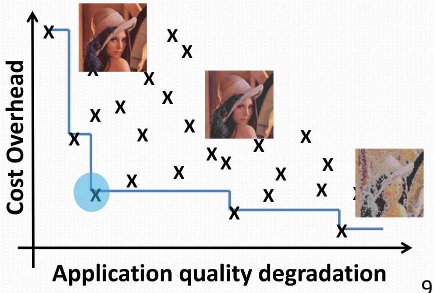
- Produce outputs of acceptable quality despite approximate computation
  - Perceptual limitations
  - Redundancy in data and/or computations
  - Noisy inputs
- Digital communications, media processing, data mining, machine learning, web search, …


Input
K-Means Clustering
Fully Correct Computations
10% Erroneous Computations
5% Erroneous Computations

e.g. Image Segmentation

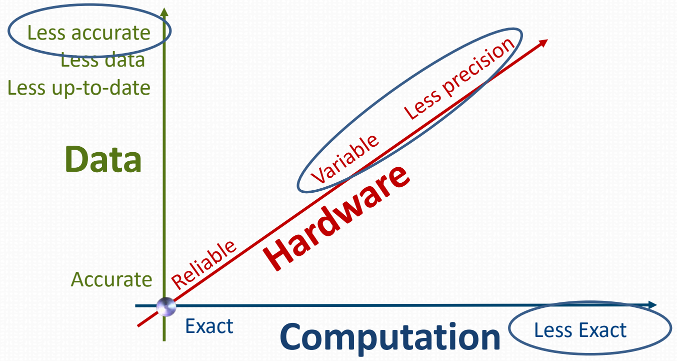
Approximate Computing

- Play with **approximations** to reduce **energy** and increase execution speed while keeping **accuracy in acceptable limits**
  - Relaxing the need for fully precise operations
- Design-time/run-time
- Abstraction levels



Approximate Computing

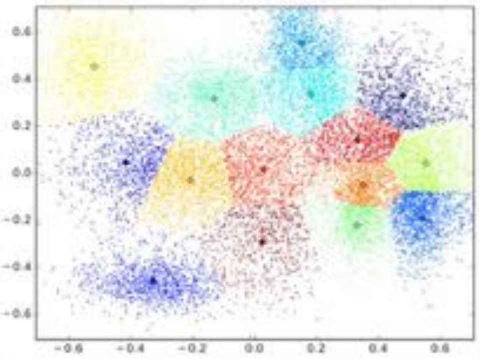
- Three dimensions to explore



Note: Precision (#bits) ≠ Accuracy (quality)

K-Means Clustering

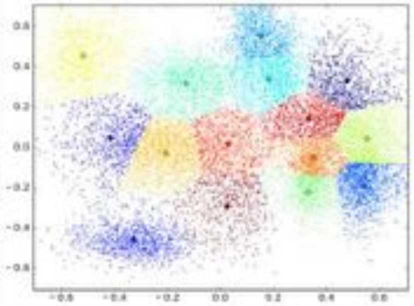
- Data mining, image classification, etc.
- A multidimensional space is organized as:
  - *k* clusters S_i,
  - S_i defined by its centroid μ_i



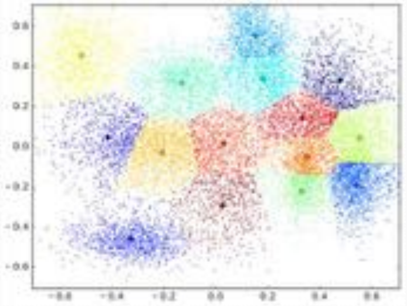
- Finding the set of clusters $S = \{S_i\}_{i \in [0,k-1]}$ satisfying $\arg\min\limits_{S} \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2$ is NP-hard (solved here by Lloyd's iterations)

Approximate K-Means Clustering

- W = 16 bits, accuracy = 10^{-4}
- No major (visible) difference with reference



Reference: double

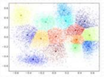


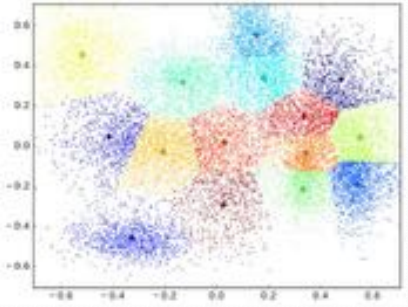
Floating-point: ct_float$_{16}$

5-bit exponent
11-bit mantissa

Approximate K-Means Clustering

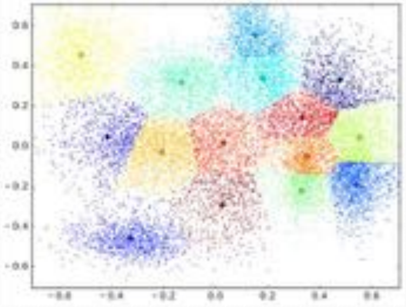
- W = 16 bits, accuracy = 10^{-4}
- No major (visible) difference with reference





Fixed-Point: ac_fixed$_{16}$

3-bit integer part
13-bit fractional part

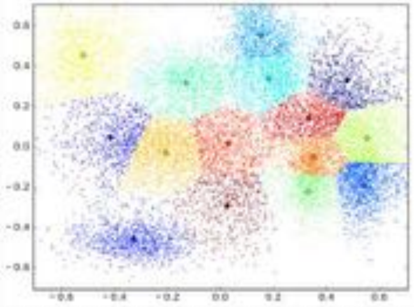


Floating-point: ct_float$_{16}$

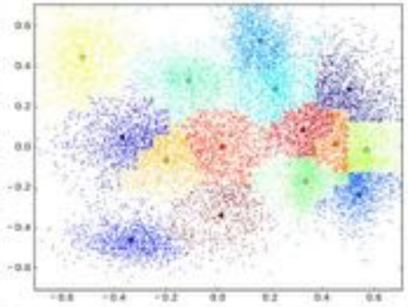
5-bit exponent
11-bit mantissa

Approximate K-Means Clustering

- W = 8 bits, accuracy = 10^{-4}
- 8-bit float is still practical



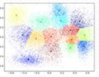
Reference: double



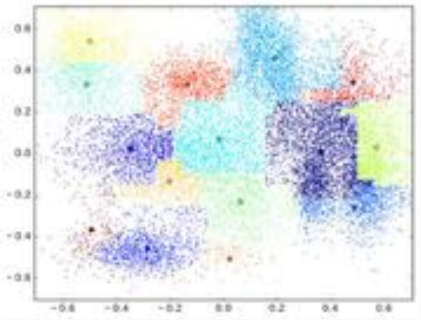
Floating-Point: ct_float$_{8}$

5-bit exponent
3-bit mantissa

Approximate K-Means Clustering

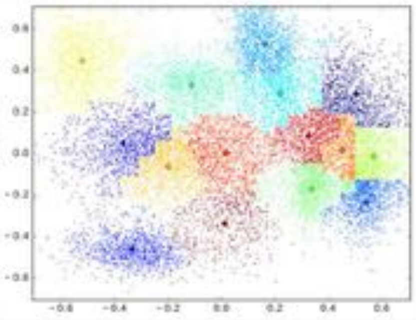


- W = 8 bits, accuracy = 10^{-4}
- 8-bit float is better and still practical



Fixed-Point: ac_fixed$_8$

3-bit integer part
5-bit fractional part

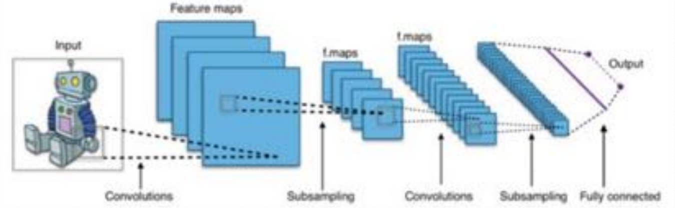


Floating-Point: ct_float$_8$

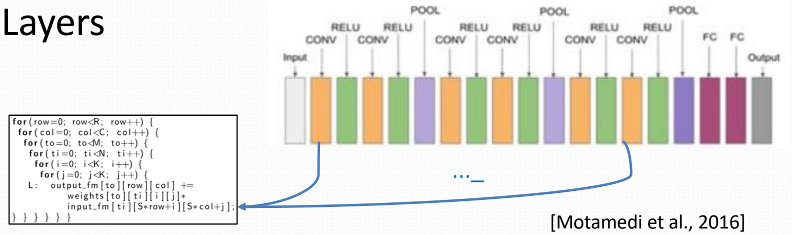
5-bit exponent
3-bit mantissa

Deep Convolutional Neural Networks

- General organization



- Layers



```
for(row=0; row<R; row++) {
 for(col=0; col<C; col++) {
  for(to=0; to<M; to++) {
   for(ti=0; ti<N; ti++) {
    for(i=0; i<K; i++) {
     for(j=0; j<K; j++) {
   L:   output_fm[to][row][col] +=
          weights[to][ti][i][j]*
          input_fm[ti][S*row+i][S*col+j];
} } } } } }
```

[Motamedi et al., 2016]

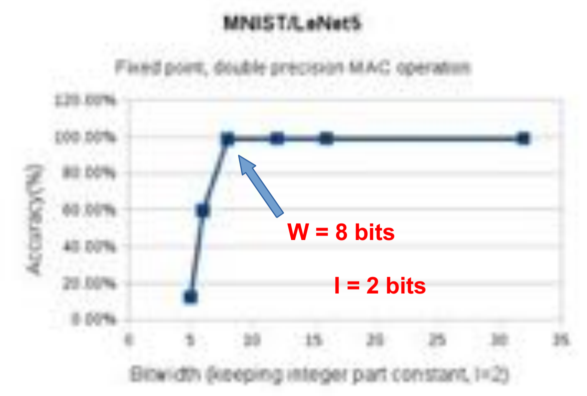
Resilience

Aoccdrnig to a rscheearch at Cmabrigde Uinervtisy, it deosn’t mttaer in waht oredr the ltteers in a wrod are, the olny iprmoatnt tihng is taht the frist and lsat ltteer be at the rghit pclae. And we spnet hlaf our lfie larennig how to splel wrods. Amzanig, no!

- Our biological neurons are fault tolerant to computing errors and noisy inputs

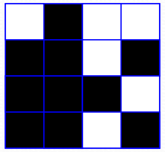
Approximate CNNs

- 10k images, MNIST/Lenet
- Fixed-Point Arithmetic



Summary

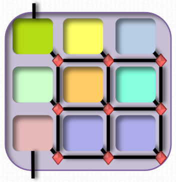
- Energy consum
  - True in embed
  - True in HPC, n
- End of Moore’s law…
- Multicores but utilization wall
  - Percentage of a chip that can switch at full frequency drops exponentially

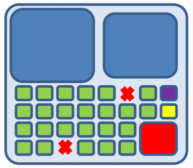


Dark Silicon

What’s next?

- Dark Silicon is also an opportunity
  - Heterogeneous manycore architectures

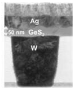




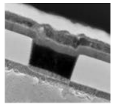
- Efficiency of hardware specialization
  - Domain-specific architectures and languages
- Computing just right
  - @design-time or @run- time

What’s next?

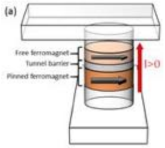
- Emerging devices
- Cells, brain, neurons have “analog” behavior
- And compute with very low precision
- Making neuromorphic computing more efficient



Phase Change Memory



Memristors, Oxide Resistive Memory



Spin Torque Magnetic Memory