Mathematical modeling of many-cores

Ran Ginosar

Technion, Israel

September 2013

Outline

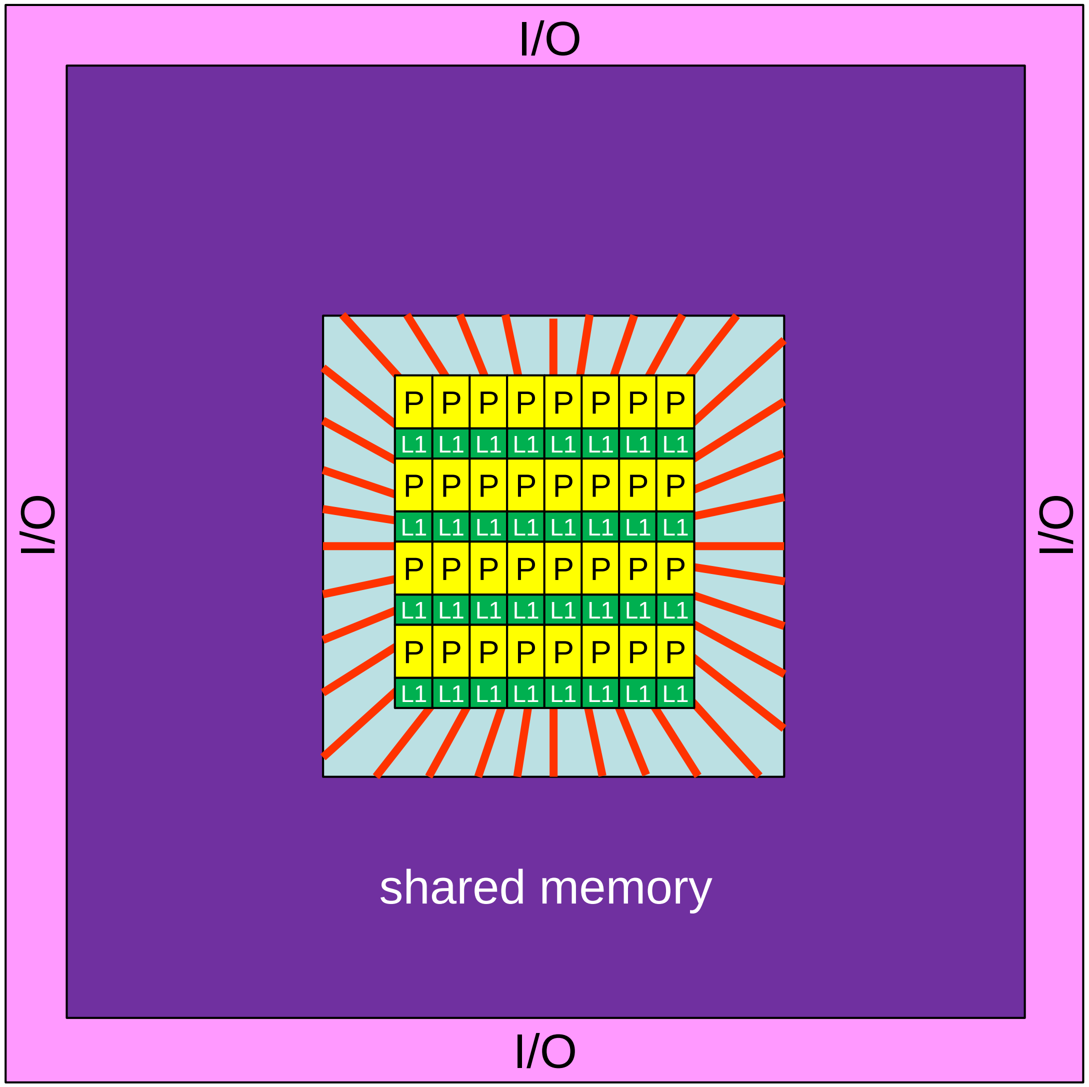
- Many core architectures
- Mathematical model
- Open questions

define many-cores

- Many-core is:
  - a single chip
  - with many (how many?) cores and on-chip memory
  - running one (parallel) program at a time, solving one problem
  - an accelerator
- Many-core is NOT:
  - Not a “normal” multi-core
  - Not running an OS
- Contending many-core architectures
  - Shared memory (the Plural architecture, XMT)
  - Tiled (Tilera, Godson-T)
  - Clustered (Rigel)
  - GPU (Nvidia)
  - SIMD
  - Associative Processor
- Contending programming models

Five many-core architectures

Shared Memory Manycore



EXAMPLE

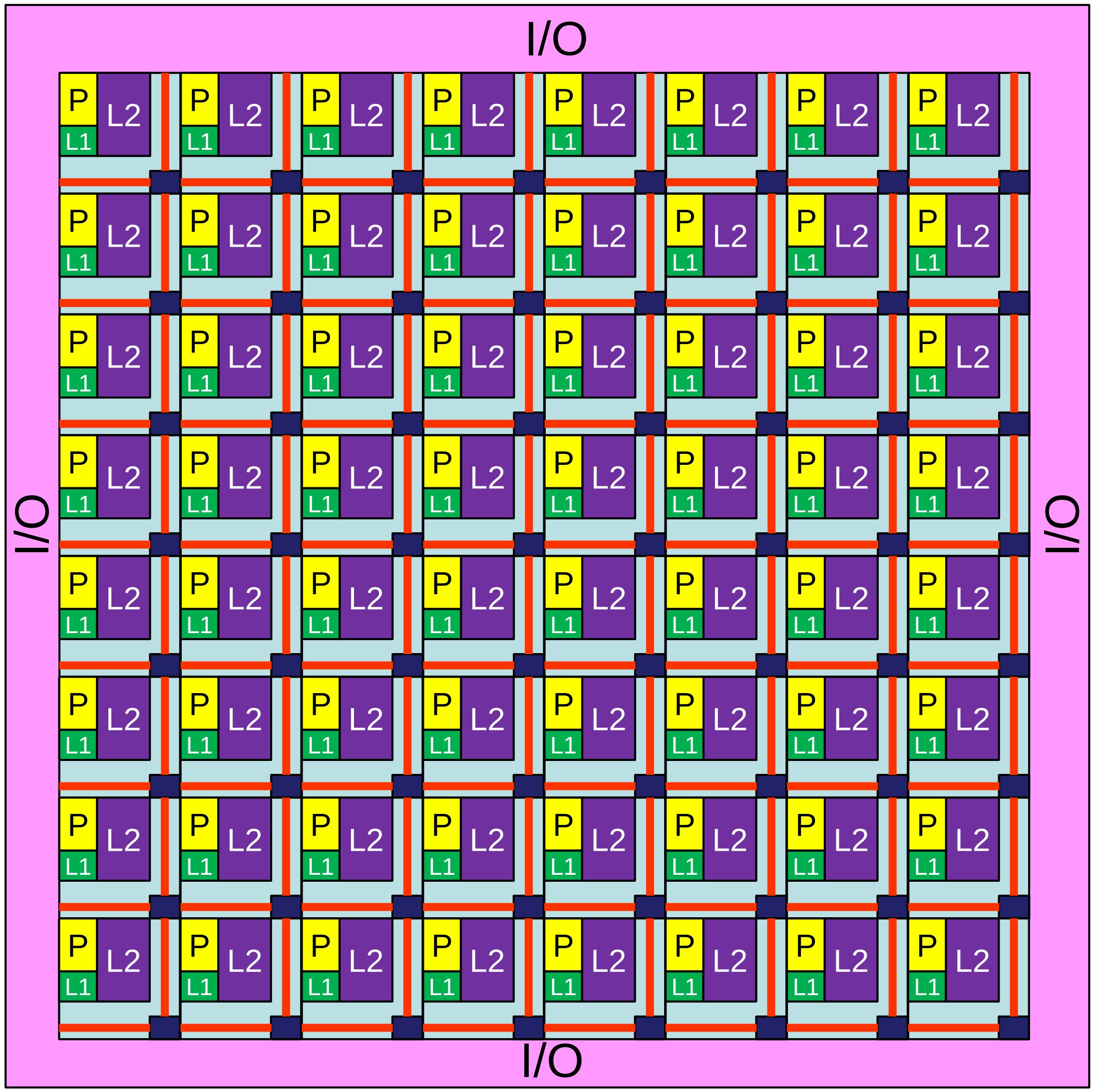
64 cores

64 L1 caches
16kB x64
= 1 MB

Core-to-mem
network

shared mem
many banks
1 MB x256
= 256 MB

Tiled Manycore



EXAMPLE

64 tiles

64 L1 caches
16kB x64
= 1 MB

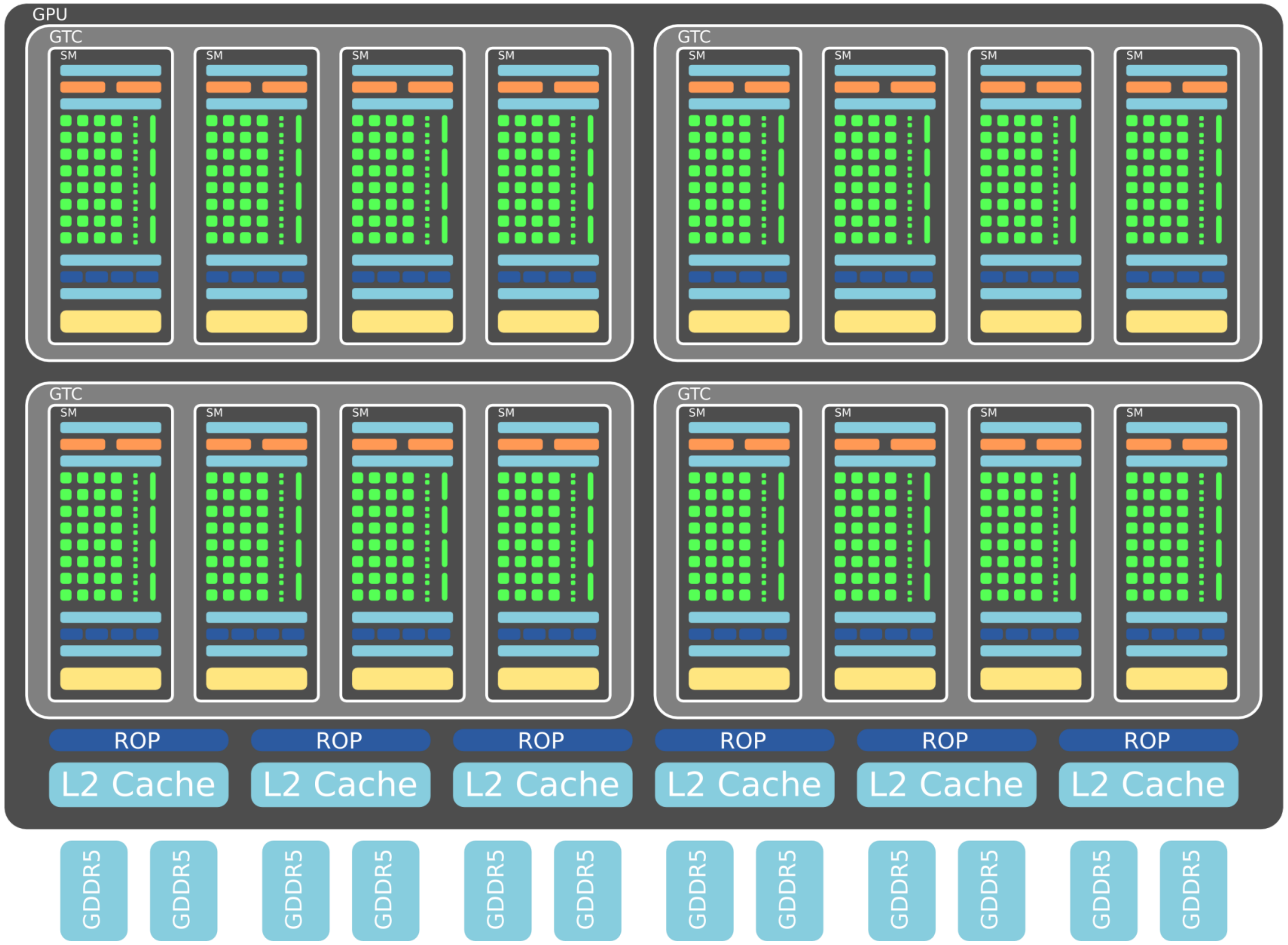
4 MB L2 x64
= 256 MB

mesh NOCs

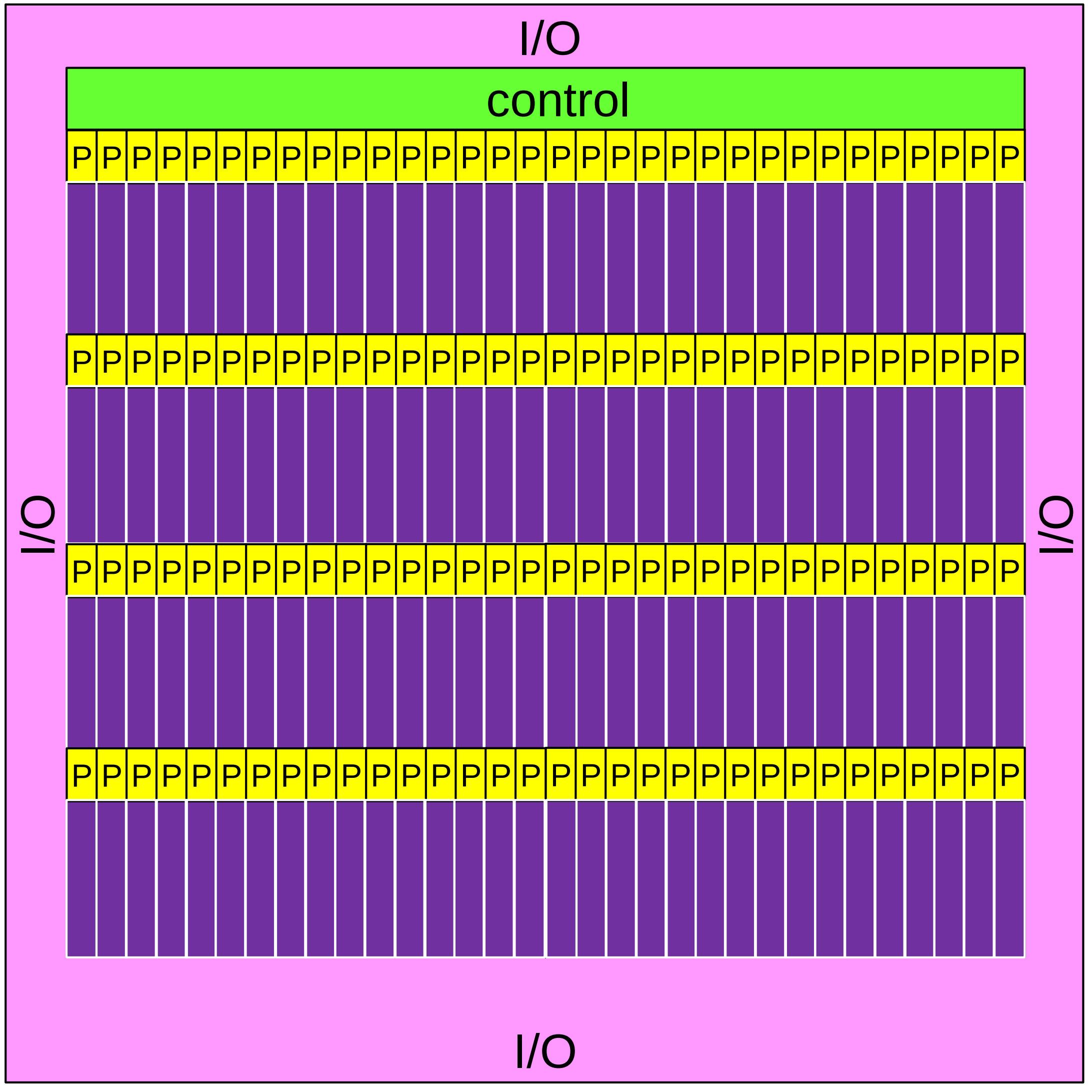
Directory:
All L2s = L3

GPU

EXAMPLE



SIMD

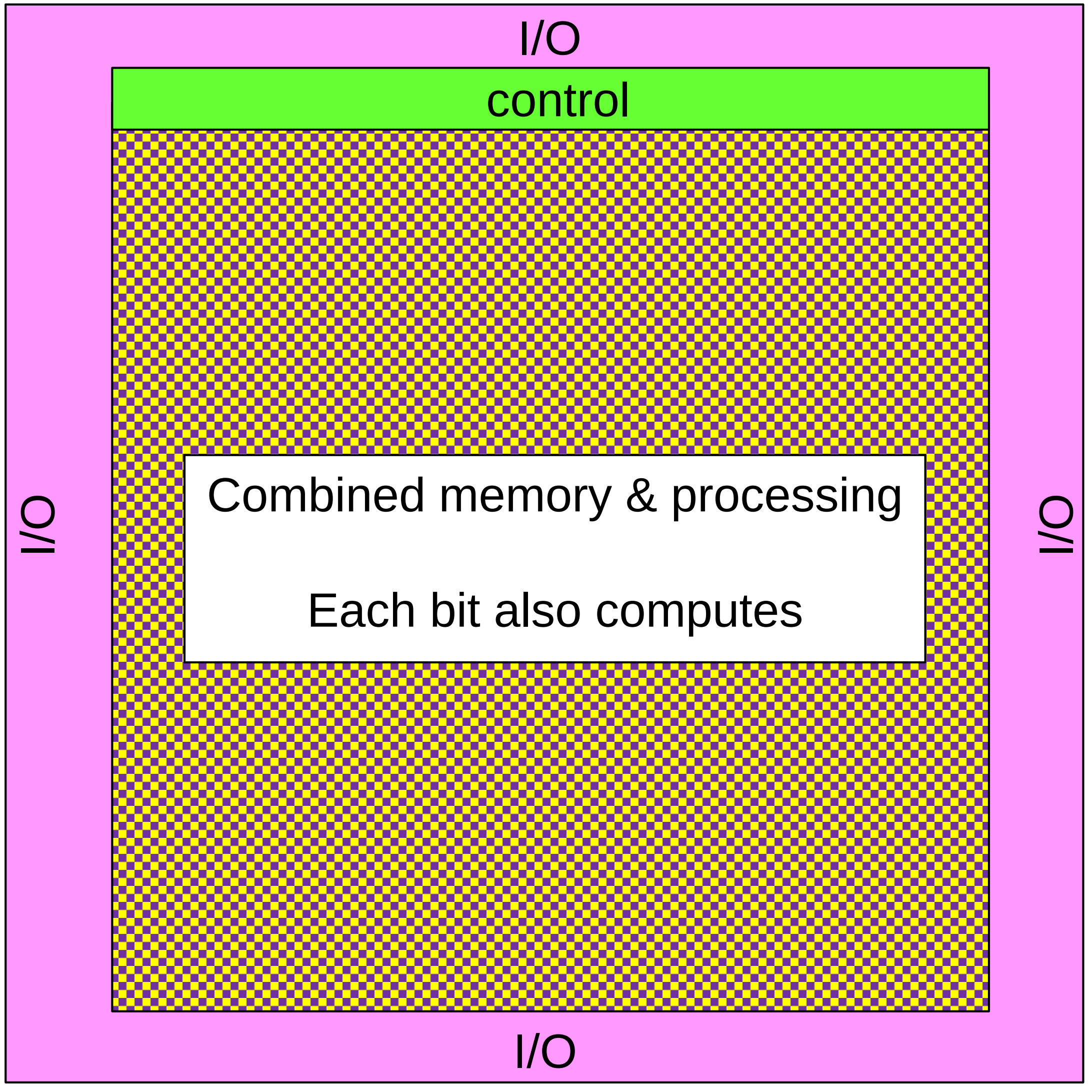


EXAMPLE

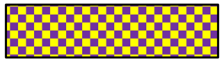
256 cores

memory banks
1 MB x256
= 256 MB

Associative Processor



EXAMPLE



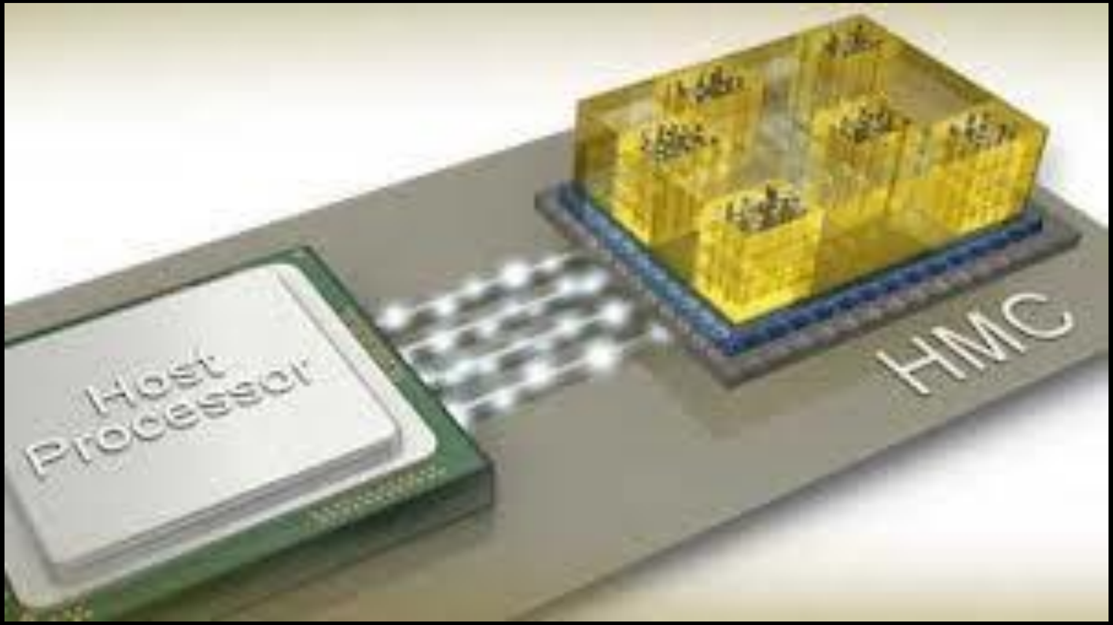
128 MB

(each bit twice larger than SRAM)

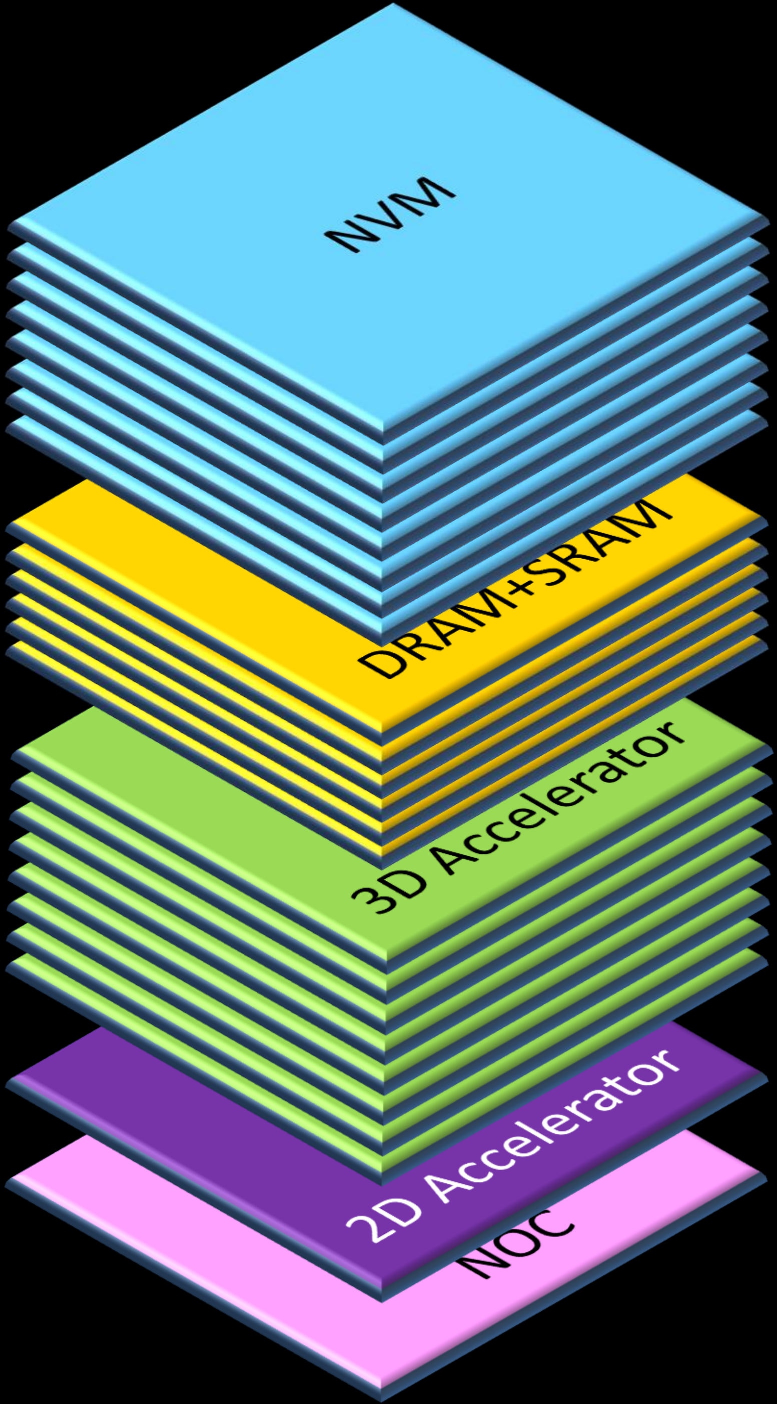
3D

Add 3D logic & 3D memory

- The HMC industry already makes the first step



- 100,000 TSV vertical interconnects



Store & Compute

- 1 Tbyte / chip in 2020
  - Combined DRAM + NVM
- Many many-cores
- NoC & 3D NoC
- Must be low power & cold: 0.1W

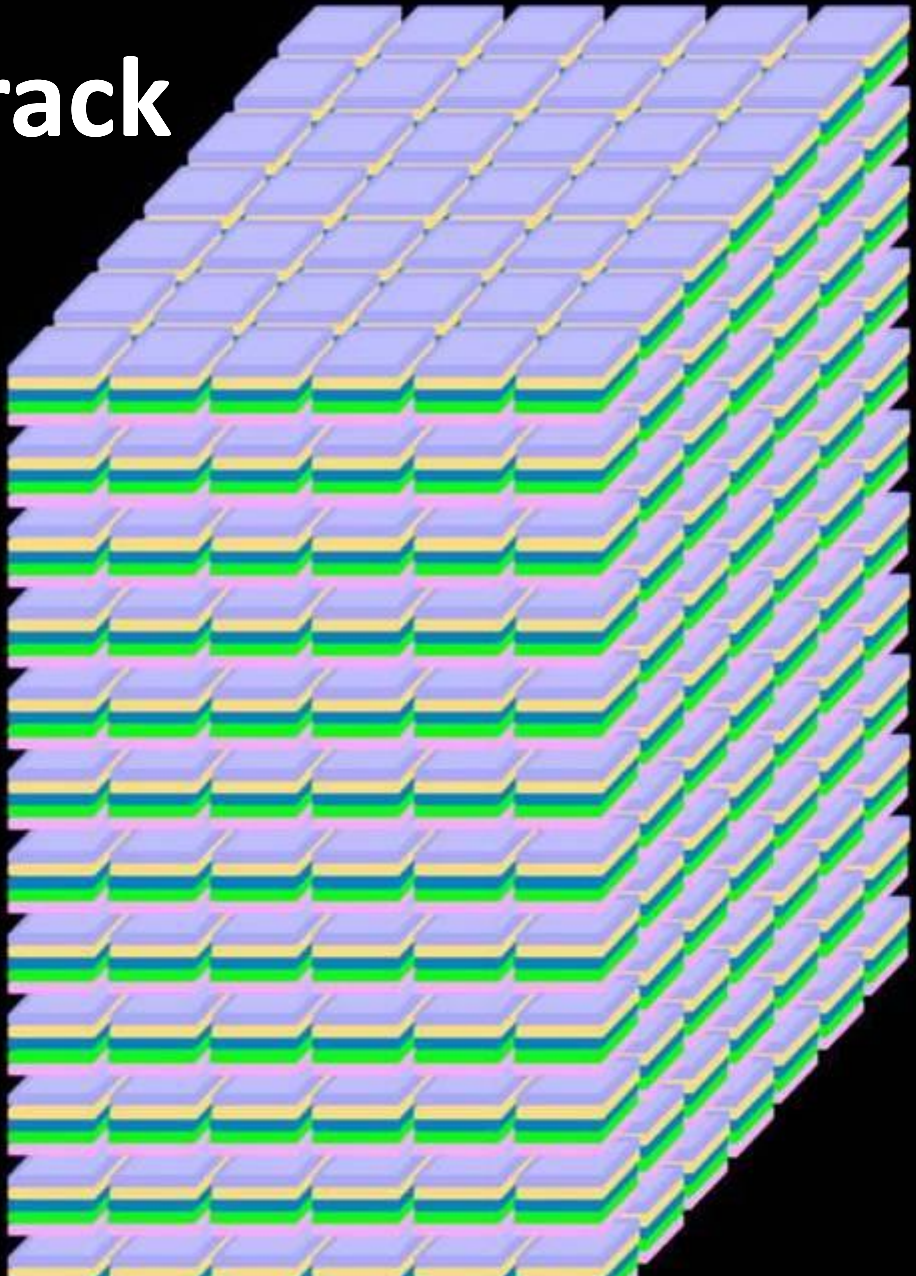
- THIS WILL CHANGE MANY-CORE ARCHITECTURE

Many cubes in a rack

- 1000 cubes
- 100 W

Many racks in a supercomputer

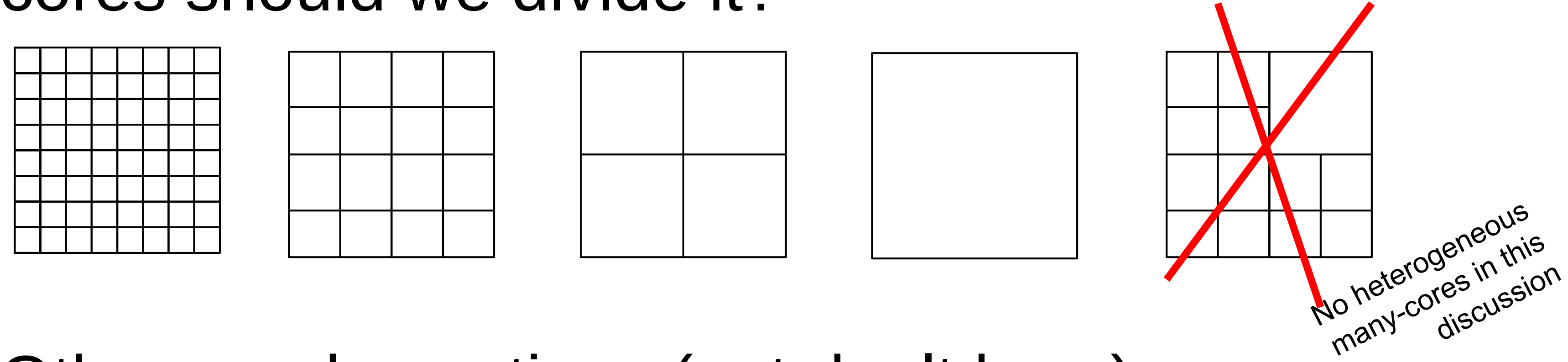
- Less than 1 MW



Modeling Many-core Architectures

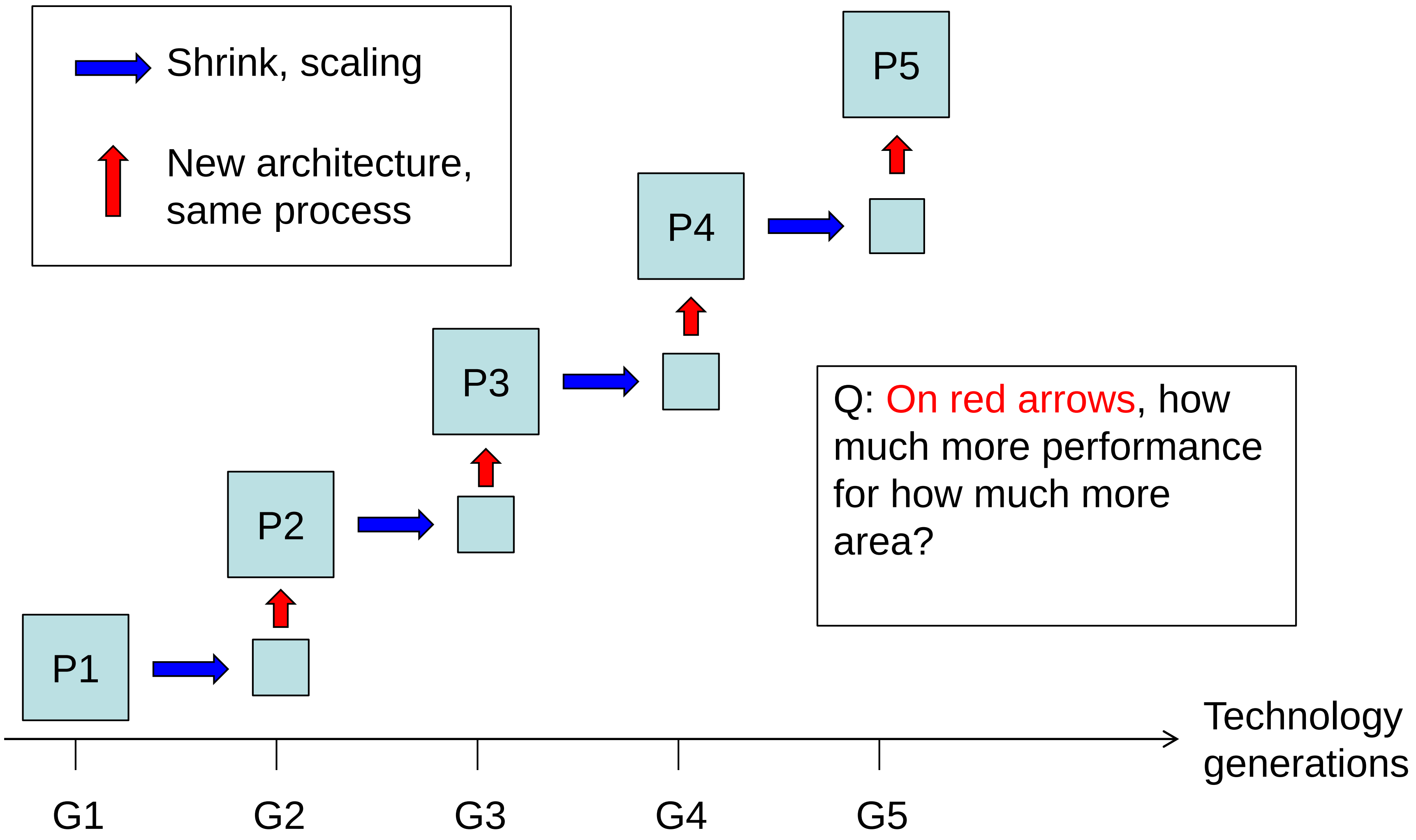
A first many-core research question

- Given fixed ***area***, into how many processor cores should we divide it?



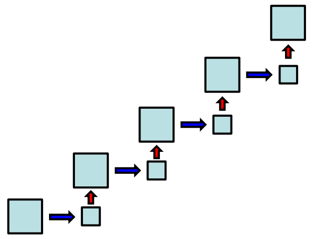
- Other good questions (not dealt here):
  - Given fixed ***power***, how many cores? which cores?
  - Given fixed ***energy***, how many cores? which cores?
  - Given target performance, how many? Which?
- Analysis can be based on Pollack’s rule

The history at the basis of Pollack’s analysis



Q: On red arrows, how much more performance for how much more area?

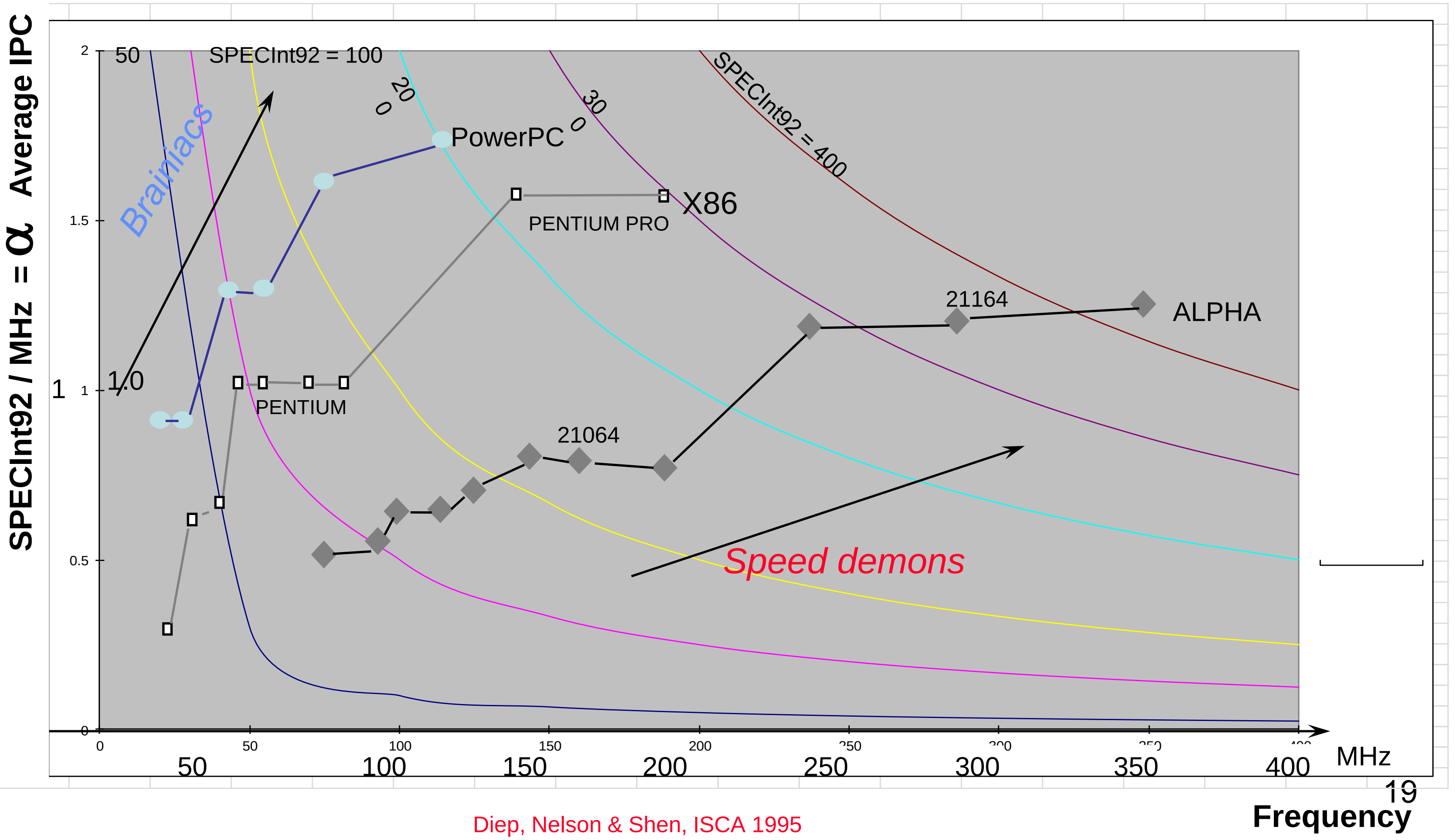
Pollack's rule for processors: Area or Power vs. Performance



- Pollack (& Borkar & Ronen, Micro 1999) observed many years of (intel) architecture
- In each Intel technology node, they compared:
  - Old µArch (shrink from previous node)
  - New µArch (faster clock and/or higher IPC)
- They noted:
  - New µArch used 2-3X larger area
  - New µArch achieved 1.5-1.7X higher performance
    - Resulting from both higher frequency and higher IPC
  - They did not consider power increase
    - Who thought about power in 1999?
- Observation: Performance ~ $\sqrt{area}$

Performance = IPC × Frequency

- Experience shows: for higher performance, both IPC and frequency must be increased



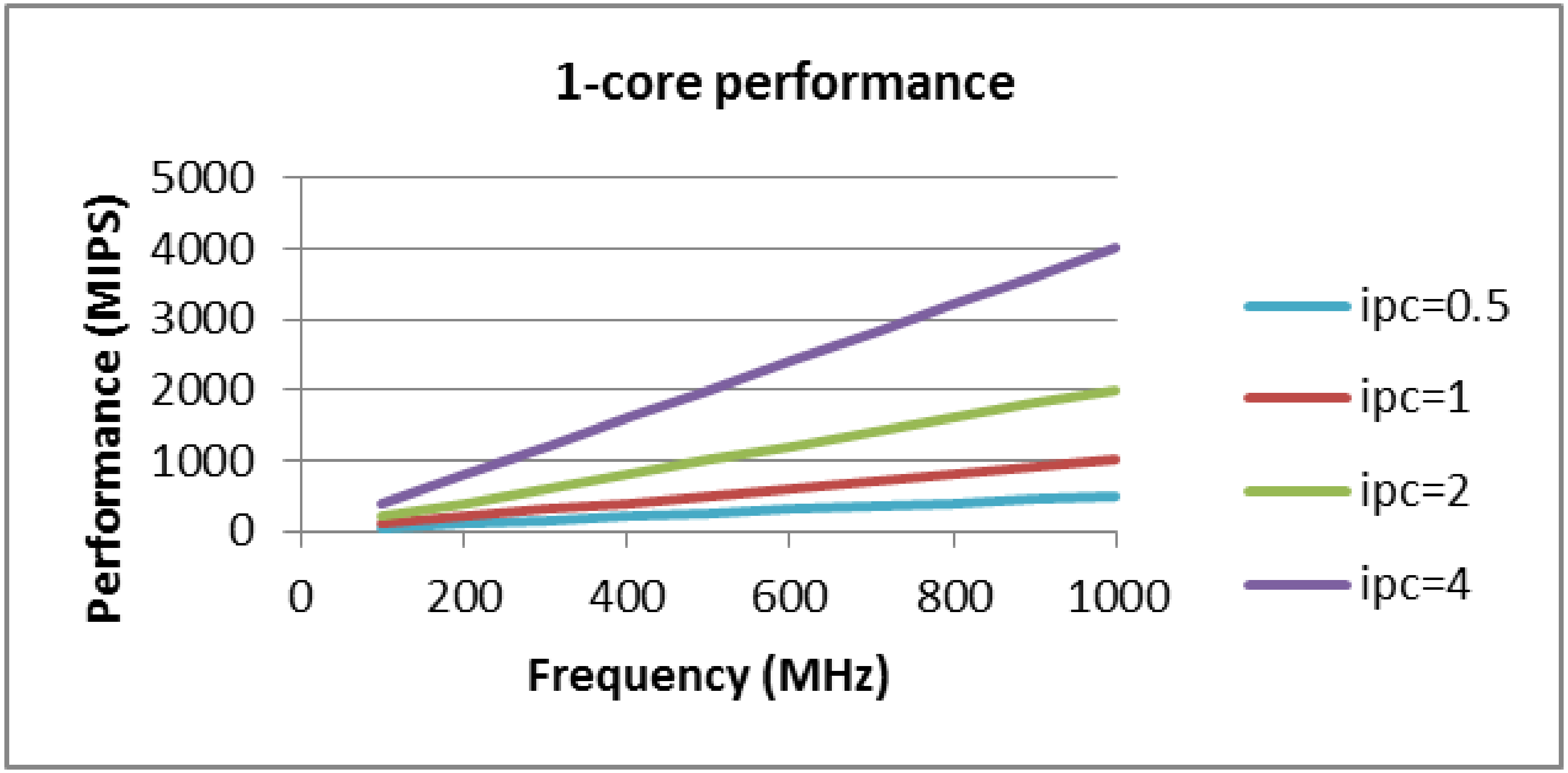
Diep, Nelson & Shen, ISCA 1995

The many-core *fixed-total-area* model

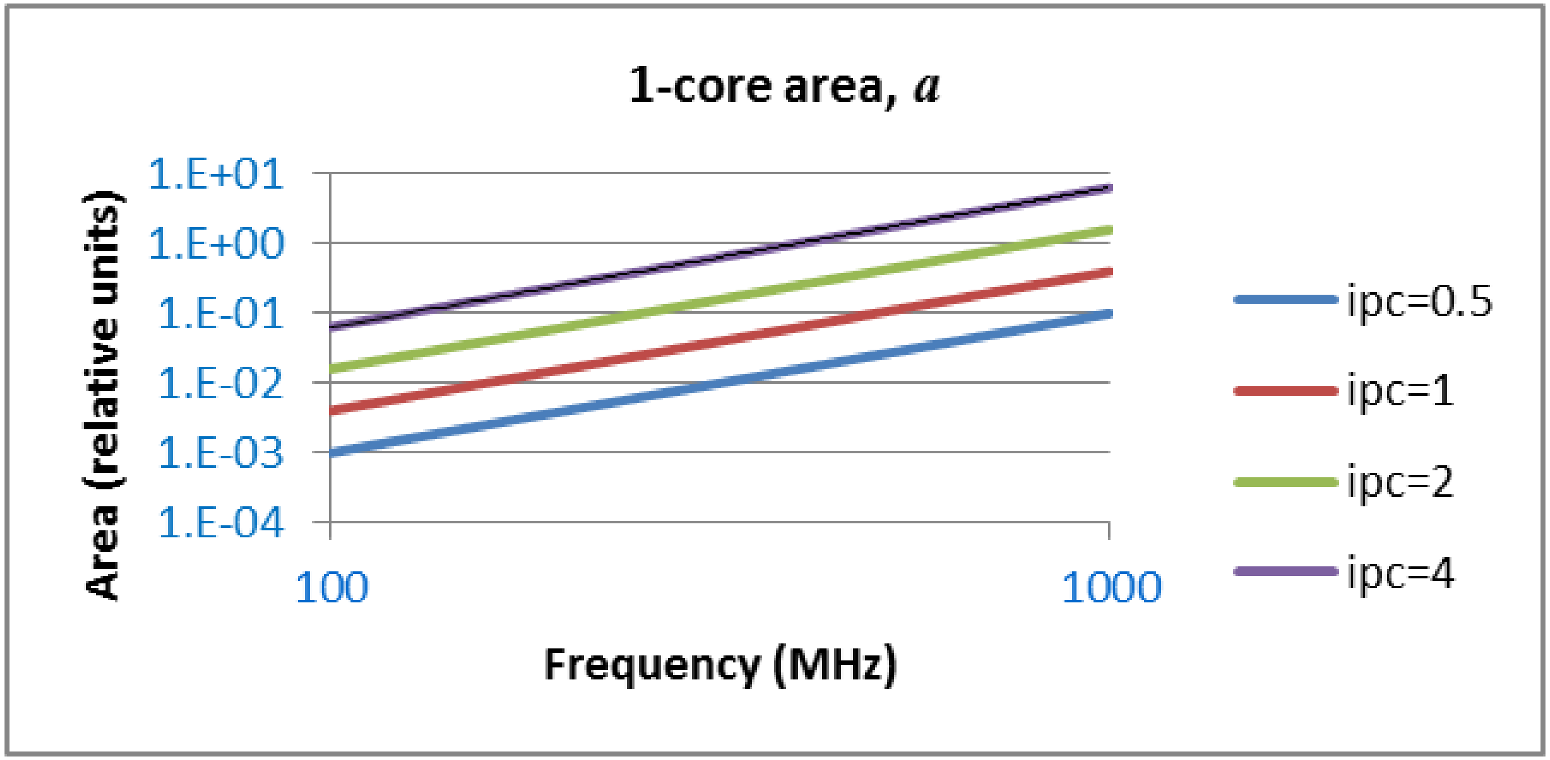
- Assume fixed chip area (typically 300-500 mm^2)
- Split chip area A = A_{cores} + A_{mem}
  - Split (memory size) affects on-chip hit rate
  - A_{mem} may be further split into A_{L1}+A_{L2}
- Divide A_{cores} into m cores. <u>How many ?</u>
  - Area of each core: $a = \frac{A_{cores}}{m}$. $\quad$ $Thus,\ m \sim {}^{1}/_{a}$
- [Pollack's]: core area determines core performance. Select *IPC* and frequency f so that:
  - $Performance\ (core) = IPC \times f \sim \sqrt{a}.\quad Thus,\ a \sim IPC^2 f^2\ ,\ m \sim {}^{1}/_{IPC^2 f^2}$
  - $Power\ (core) \sim a \times f \sim IPC^2 f^3$
- Assume perfect parallelism (at least as upper bound)
  - $Performance\ (m\ cores) = IPC \times f \times m \sim \frac{IPC \cdot f}{IPC^2 f^2} = \frac{1}{IPC \cdot f} \sim \frac{IPC \cdot m}{IPC\sqrt{m}} = \sqrt{m}$
  - $Power\ (m\ cores) = a \times f \times m \sim \frac{IPC^2 f^3}{IPC^2 f^2} = f \sim \frac{1}{IPC\sqrt{m}}$

$$\boxed{\text{Summary: Performance} \sim \frac{1}{f} \sim \sqrt{m}, \qquad \text{Power} \sim \frac{1}{\sqrt{m}} \sim f, \qquad m \sim \frac{1}{f^2}}$$

$Performance\ (core) = IPC \times f$

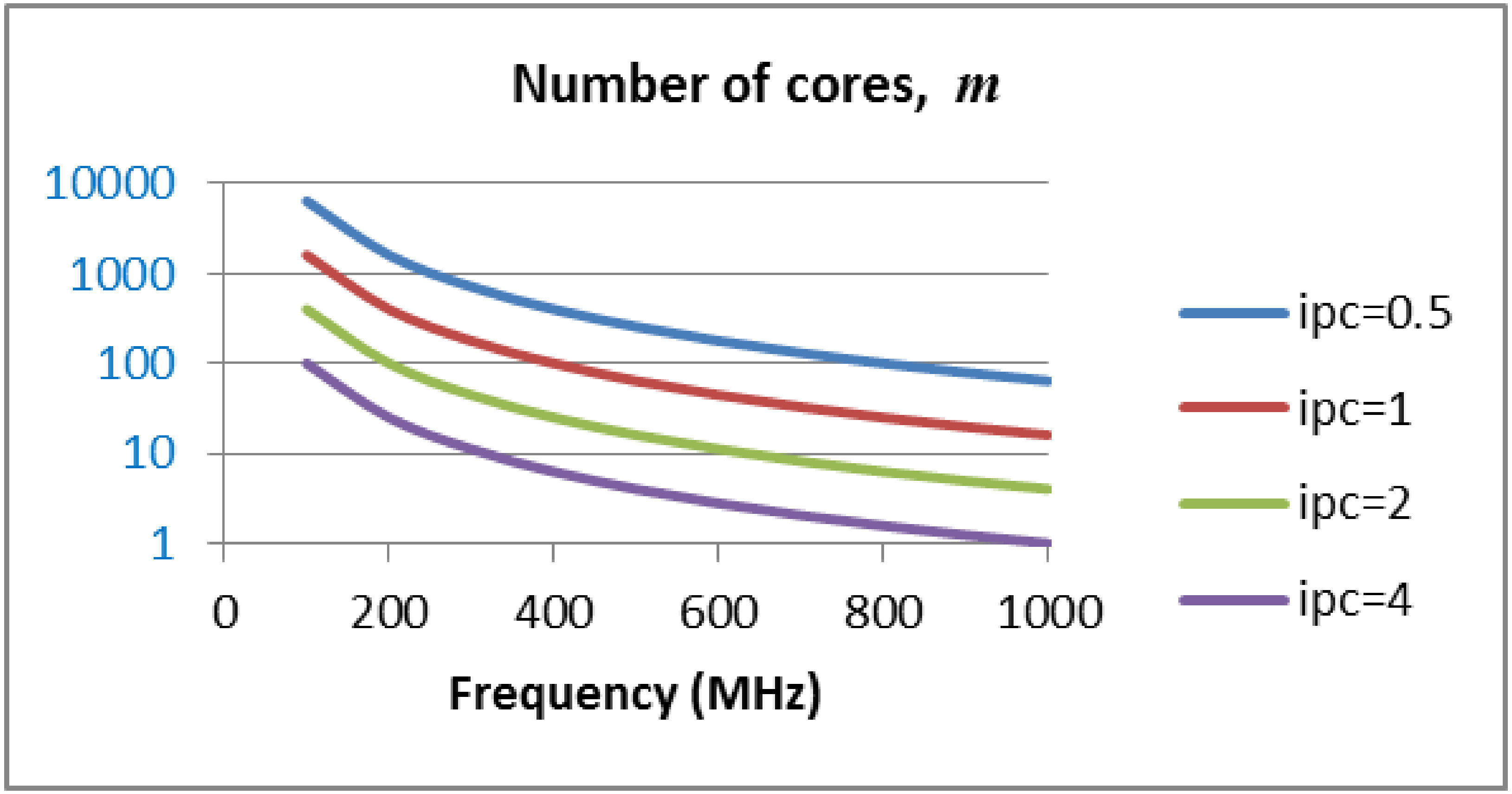


$a \sim IPC^2 f^2$



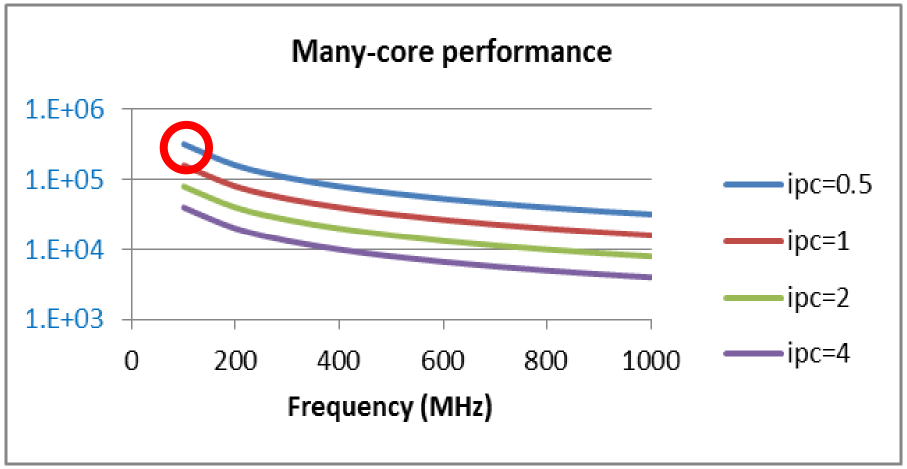
For each IPC curve, $a \sim f^2$

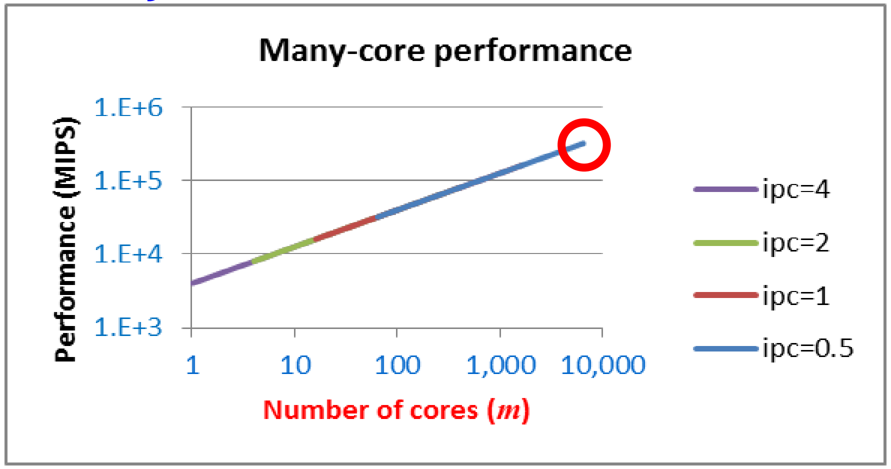
$$m \sim \frac{1}{IPC^2 f^2}$$



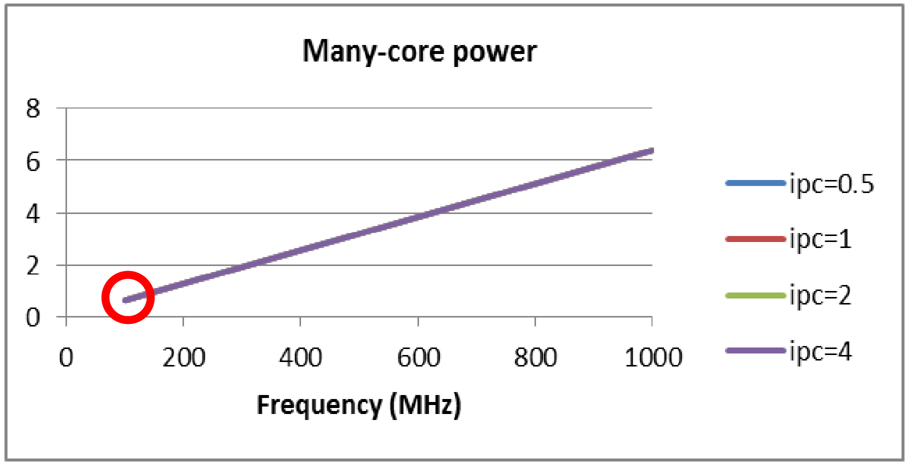
For each IPC curve, $m \sim \frac{1}{f^2}$

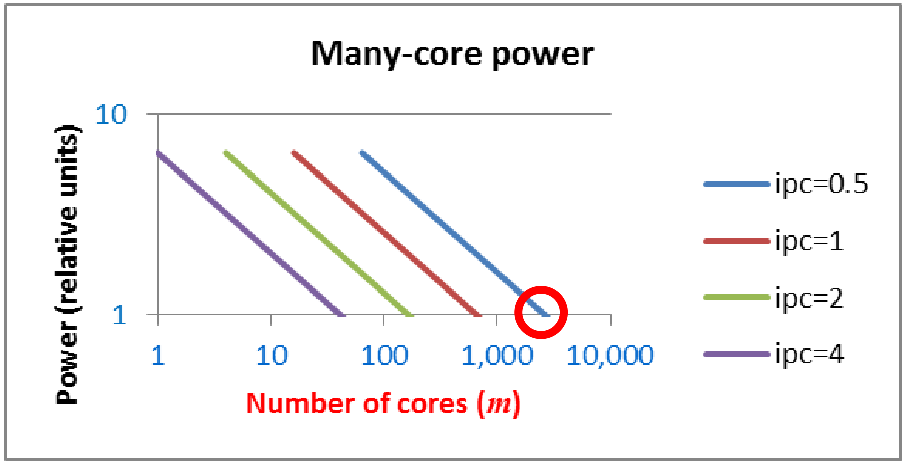
$$\text{Performance} \sim \frac{1}{f} \sim \sqrt{m}$$



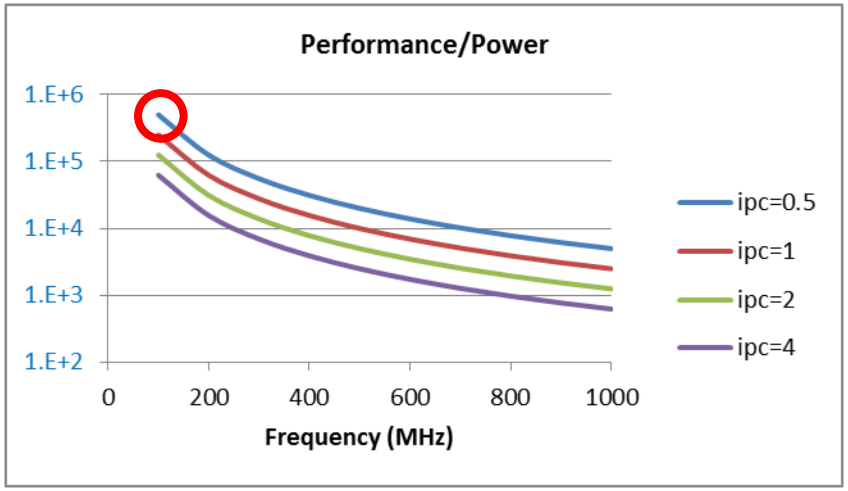


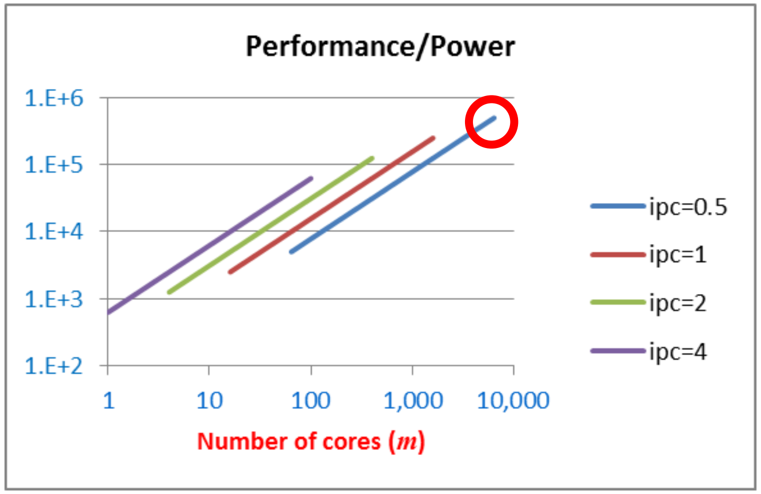
$$\text{Power} \sim f \sim \frac{1}{\sqrt{m}}$$





$$\frac{Performance}{Power} \sim \frac{1/f}{f} = \frac{1}{f^2} \sim \frac{\sqrt{m}}{1/\sqrt{m}} = m$$





Analysis of the results so far:

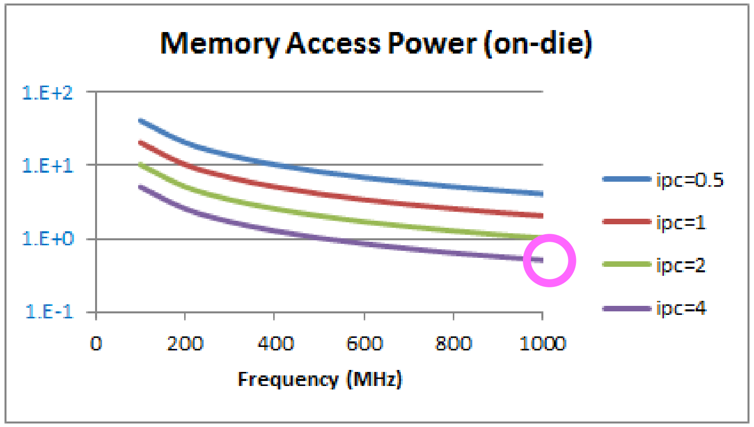
- Slower frequency and lower IPC → higher performance, lower power
- Thanks to Pollack's square rule

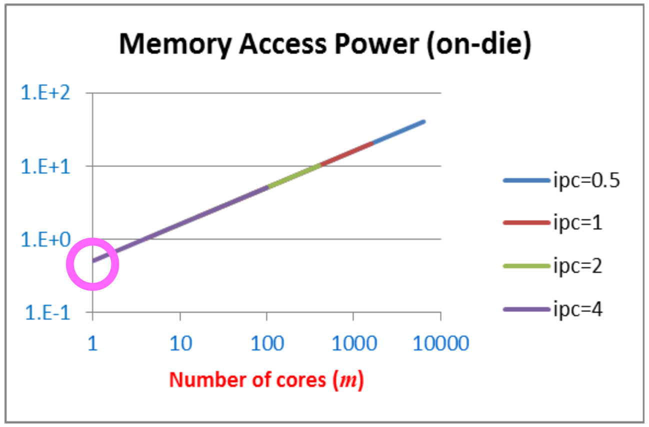
But this changes when we also consider memory power…

Now add memory

- So far, only computing power
  - Including power to access local cache/memory in each core
- But we also need to access not-so-local shared memory
- Access rate to memory: once every r_m instructions
  - E.g. about every 20 instructions
  - Assume using only on-chip memory
- Need to add memory access power to the computing power
  - Relative energy: assume access is 10x higher than exec.

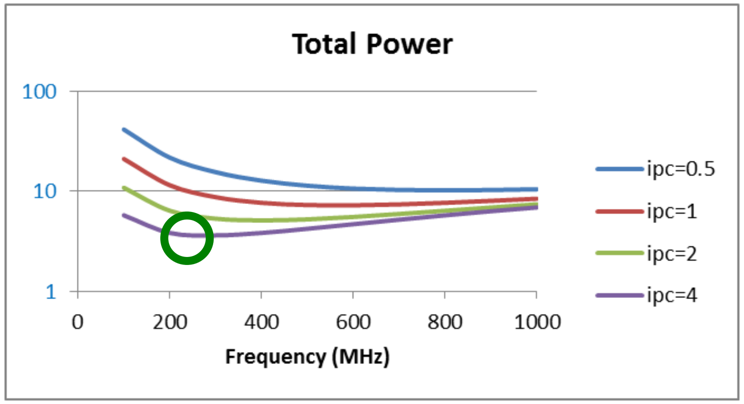
$$\frac{1}{f}$$

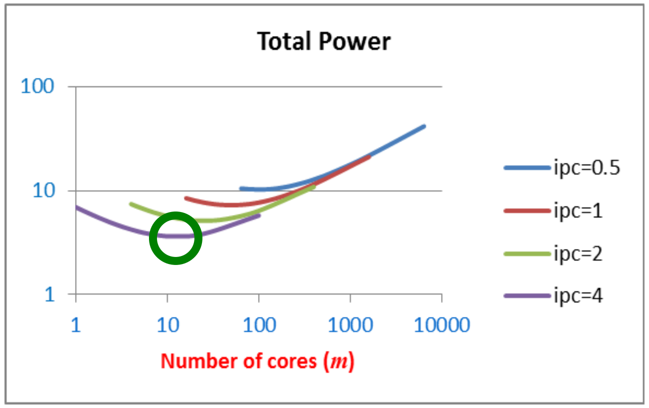




$$\sqrt{m}$$

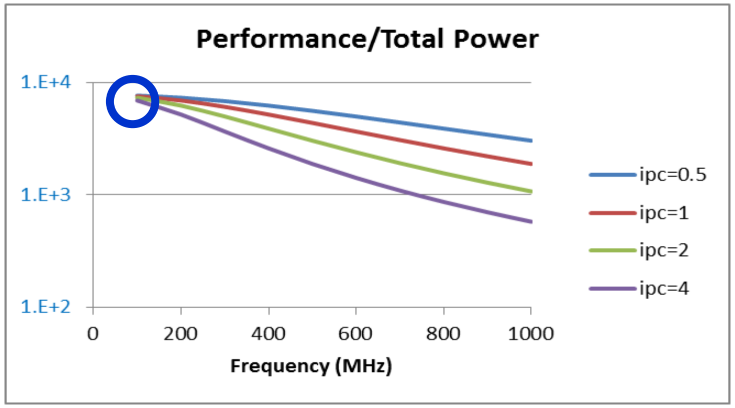
$$\frac{1}{f}+f$$

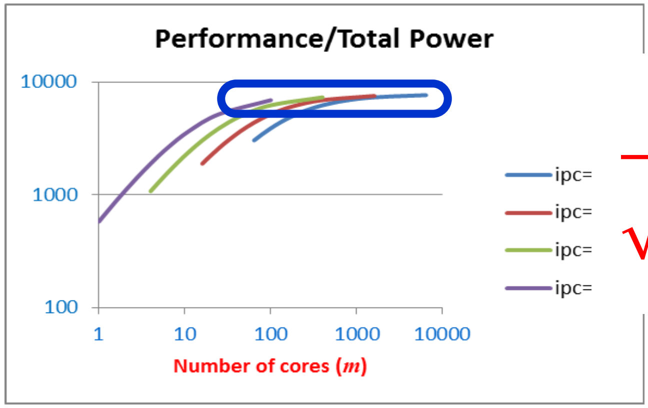




$$\sqrt{m}+\frac{1}{\sqrt{m}}$$

$$\frac{\frac{1}{f}}{\frac{1}{f}+f}$$





$$\frac{\sqrt{m}}{\sqrt{m}+\frac{1}{\sqrt{m}}}$$

Does the model apply to different architectures?

Shared memory versus Tiled architectures

Architecture	Shared memory	Tiled
Local memory	L1 in each core	L1 & L2 in each core
Global (on-chip) memory	Shared memory	L2 of other cores
Core-to-global-memory network	Dedicated cores-to-memories, e.g. MIN	Indirect via other cores/routers, e.g. mesh
Access rates (strongly depends on app; Examples:)	1/20 to L1 1/1,000 to shared mem	1/20 to L1 1/1,000 to L2 1/50,000 to others
Access time (cycles)	2 to L1 10 to shared memory	2 to L1 10 to L2 100 to others
Access energy (relative to one register instruction)	2x to L1 20x to shared mem	2x to L1 5x to L2 100x to others

Shared memory & Tiled architectures versus others

Architecture	Shared memory	Tiled	GPU	SIMD	AP
Local memory	L1 in each core	L1 & L2 in each core			
Global (on-chip) memory	Shared memory	L2 of other cores			
Core-to-global-memory network	Dedicated cores-to-memories, e.g. MIN	Indirect via other cores/routers, e.g. mesh			
Access rates (strongly depends on app; Examples:)	1/20 to L1 1/1,000 to shared mem	1/20 to L1 1/1,000 to L2 1/50,000 to others			
Access time (cycles)	2 to L1 10 to shared memory	2 to L1 10 to L2 100 to others			
Access energy (relative to one register instruction)	2x to L1 20x to shared mem	2x to L1 5x to L2 100x to others			

Summary of the model

- Considering only cores, *fixed-total-area* model implies: for highest performance and lowest power, use
  - smallest / weakest cores (lowest IPC)
  - lowest frequency
- Adding on-chip access to memory leads to a different conclusion: for lowest power and highest performance/power ratio, use
  - Strongest cores (high IPC)
  - But stay with lowest frequency
    - Lower frequency → lower access rate to global memory
- How does this apply to other architectures?