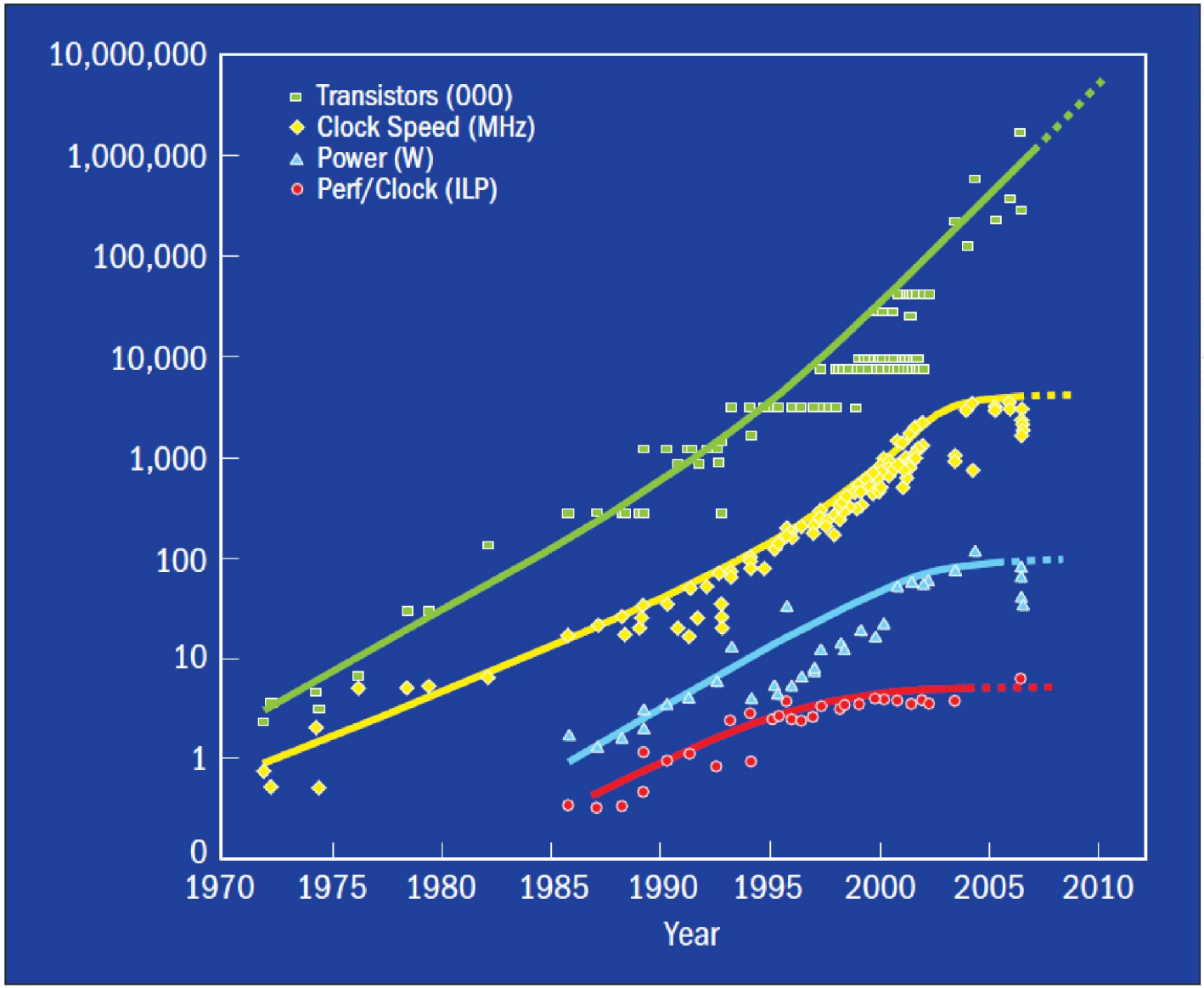
SINGULARITY UNIVERSITY
Preparing Humanity For Accelerating Technological Change

The Autumn of Moore's Law:
Scaling Up Computer Performance, 2011-2020

Jan Gray
Gray Research LLC
jsgray at acm.org

Introduction – Performance Scaling Ebbs!



[The Manycore Revolution, SciDAC Review, Fall 2009]

1. **Moore's Law:** Transistor Technology Scaling

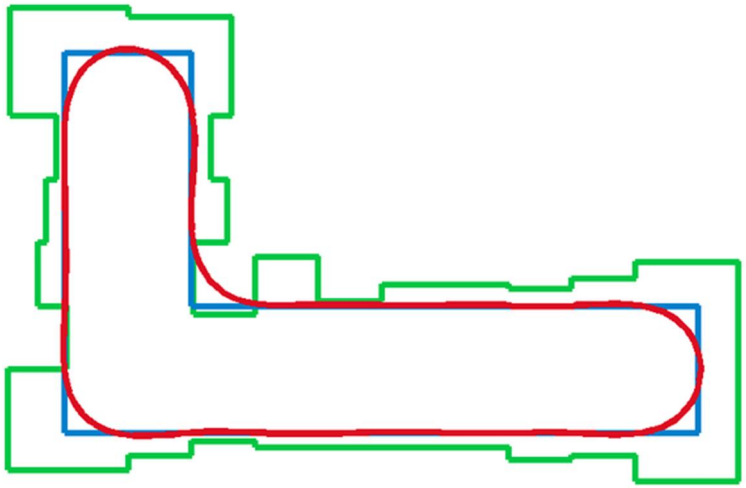
High Volume Manufacturing	2004	2006	2008	2010	2012	2014	2016	2018
Technology Node (nm)	**90**	**65**	**45**	**32**	**22**	**16**	**11**	**8**
Integration Capacity (BT)	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**
Speed (delay)	0.7	0.7	>0.7	Delay scaling will slow down				
Energy/Op	>0.35	>0.5	>0.5	Energy scaling will slow down				
Variability	Medium			High		Very High		

[from S. Borkar, Intel, Design Automation Conf., 6/07]

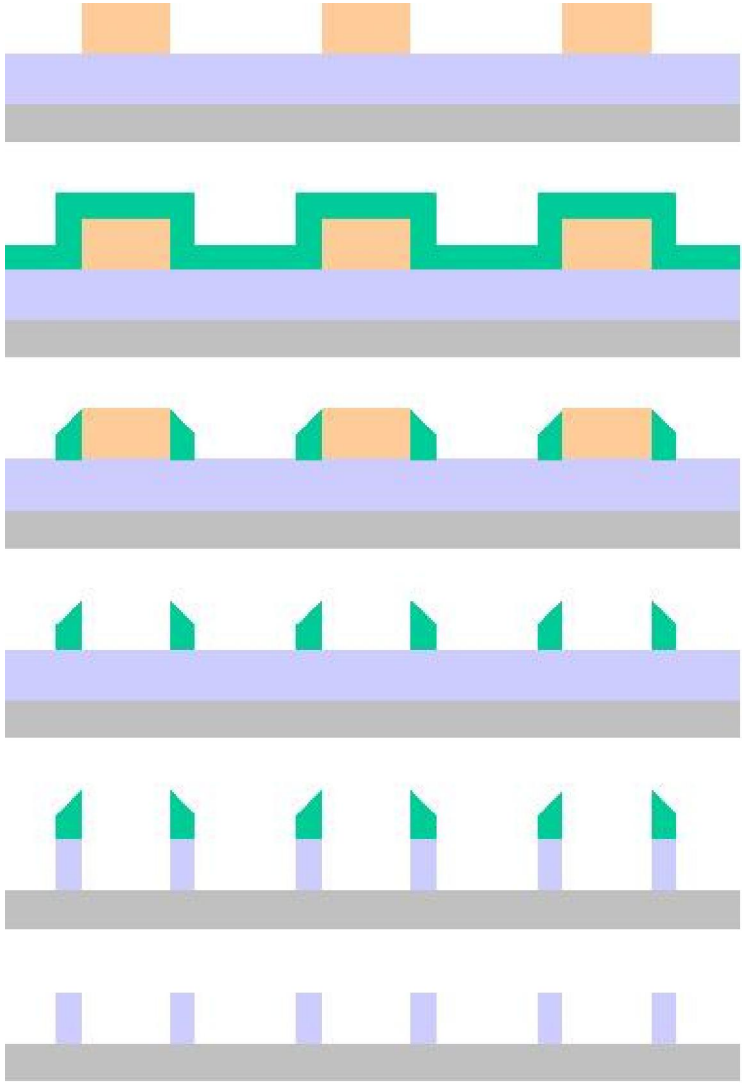
(compare to 2009 ITRS Roadmap: 2X/**2y** until 2013; 2X/**3y** after that)

Keeping Moore's Law Going 2011-20:
Tough Slogging → Manufacturing Miracles, On Schedule

- Transistor Fabrication
- Extend 193 nm UV refractive optics lithography
  - Immersion
  - Optical proximity correction
  - Double patterning



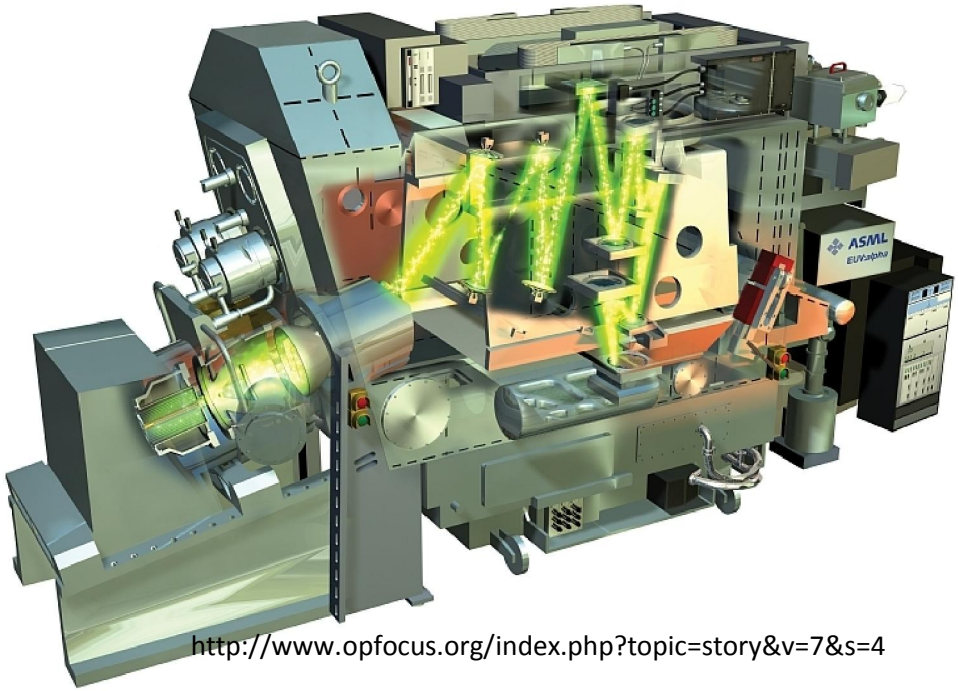
Wikipedia, OPC



Wikipedia, double patterning

Keeping Moore's Law Going 2011-20:
Tough Slogging → Manufacturing Miracles, On Schedule

- <u>Transistor Fabrication</u>
- Extend 193 nm UV refractive optics lithography
  - Immersion
  - Optical proximity correction
  - Double patterning
- Ready 13 nm Extreme UV reflective optics lithography
  - Soft X-rays, vacuum chambers
  - All new light source, mirrors, masks, resists, inspection

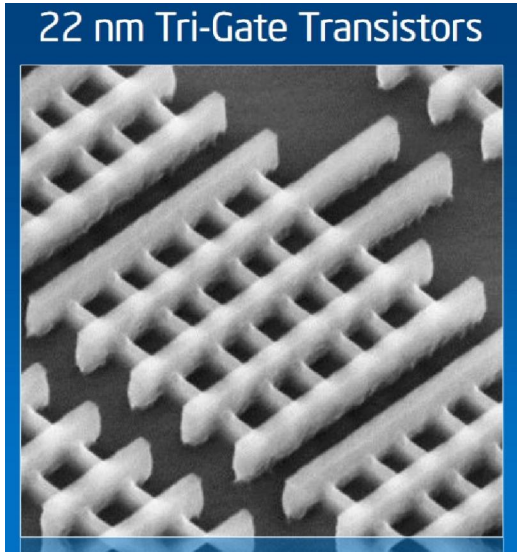


http://www.opfocus.org/index.php?topic=story&v=7&s=4

Keeping Moore's Law Going 2011-20:
Tough Slogging → Manufacturing Miracles, On Schedule

- Transistor Fabrication
- Extend 193 nm UV refractive optics lithography
  - Immersion
  - Optical proximity correction
  - Double patterning
- Ready 13 nm Extreme UV reflective optics lithography
  - Soft X-rays, vacuum chambers
  - All new light source, mirrors, masks, resists, inspection

- Transistor Evolution
- New materials in gate, insulator, channel
- New device structures



- Heroic measures, $4B fabs

Continued Memory Device Scaling?

- Charge trapping RAMs are 40 years old
  - Dynamic RAM – capacitor
  - FLASH PROM – “floating gate”
    - Only ~10s of electrons/bit …
- Go 3D? Stack cells vertically
- Resistance is futile? Phase Change RAM
  - Melt chalcogenide sites: amorphous=0, crystalline=1
  - Should scale to 5 nm cells
- Many other ideas …

As Moore's Law Ebbs…

- Slower scaling of gate delay, power; less "ideal"
- Slower transistor doublings
  - 2X/**1.5y** … 2X/**2y** … 2X/**3y** … 2X/**4y** … …
- Regular cost halvings should continue
  - Fab amortization and optimizations
- Transition to ??? – lower energy, cheaper
- When device doublings end *and* cost halvings end?
  - Pause, reflect, rethink higher level abstraction layers
  - Redoing abstractions will yield several more doublings of performance/energy

2. **Computer Architecture:** Spending Millions of Transistors On Performance

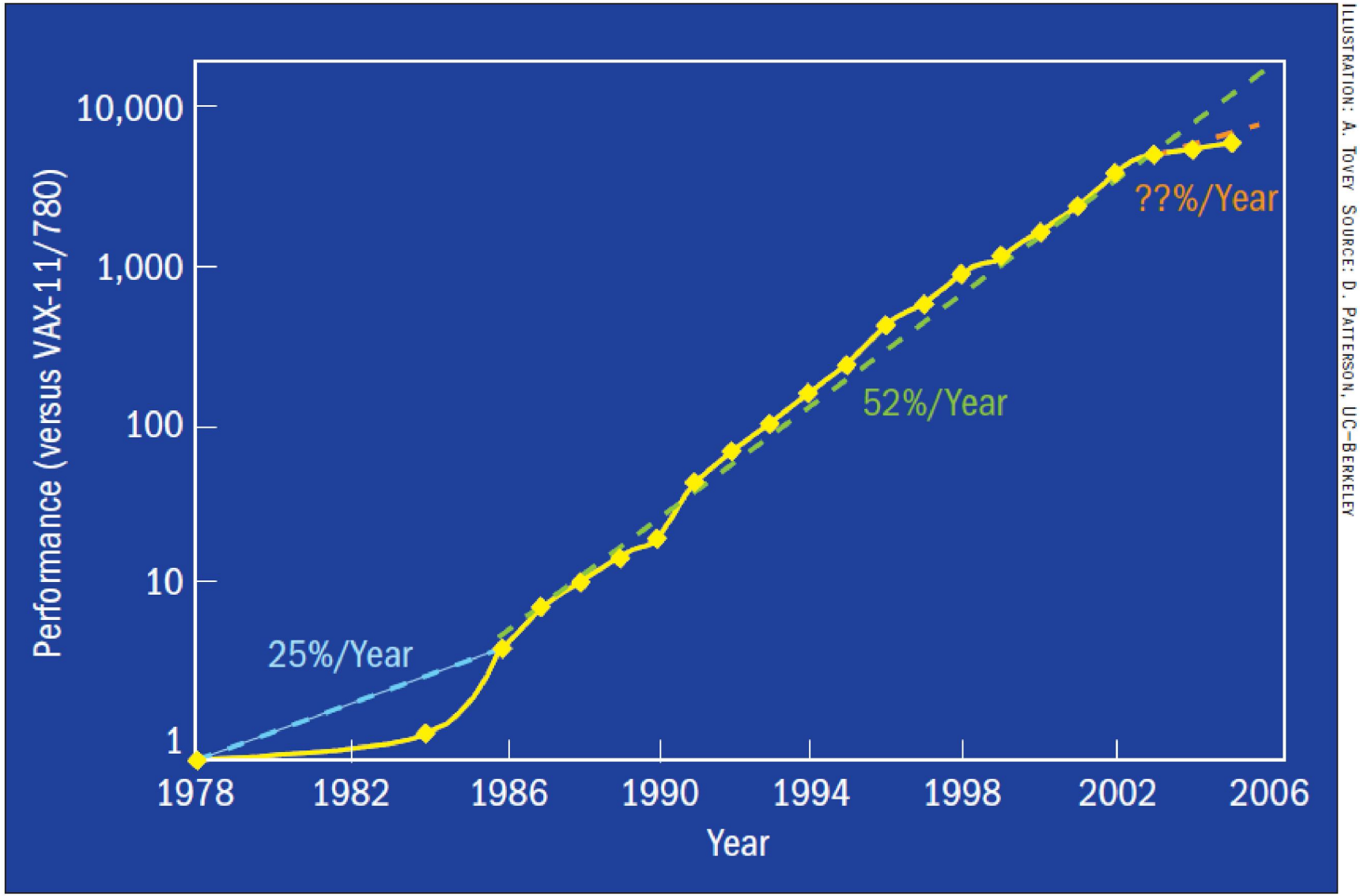


ILLUSTRATION: A. TOVEY SOURCE: D. PATTERSON, UC–BERKELEY

[The Manycore Revolution, SciDAC Review, Fall 2009]

Spending Transistors on Performance

- Goal: run old software much faster than your last chip!
- Imagine you're Henry Ford. How can you build more cars faster?
- 30 years of Instruction Level Parallelism (ILP)
  - Start with a simple slow 10,000 transistor processor. Add:
  - Richer instructions: wider integers, floating point, vectors
  - Pipelining of instruction execution
  - Memory caches
  - Multiple instruction issue and out of order execution
  - More & more → complex superfast 100M transistor processor
- "Tried every trick in the book"

Towards the Next 100X Speedup: Uniprocessor Performance Challenges

- The Memory Wall
- The Power Wall
- The Complexity Wall

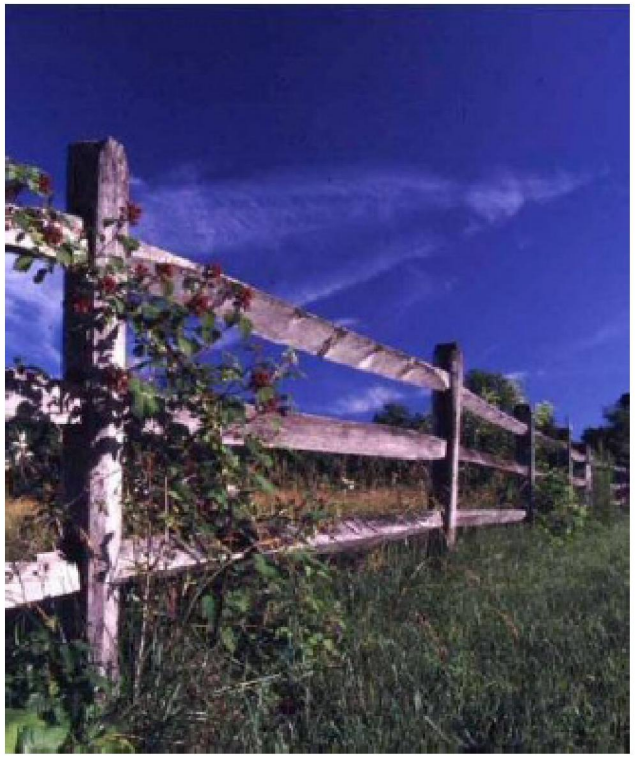
Towards the Next 100X: **The Memory Wall**

- Over 30 years…
  - CPU cycle time 1000 ns → 0.3 ns but
    DRAM access time 500 ns → 100 ns
- Cache miss stalls CPU for 100s of cycles!
- Caches and out-of-order execution help mitigate latency, at high cost in area, power, complexity

Towards the Next 100X: **The Power Wall**

- **Power ∝ CV^2f** – dynamic power of switching gates
- Over 30 years…
  - C: exponentially more, exponentially smaller transistors
  - V: 15 → 1 V *P↓225X*
  - f: 1 → 3000 MHz *P↑3000X*
  - P: 1 → **100W**
  - 10^{10} transistors, but can't switch them all, or fast
- Phones, PCs, data centers all demand *lower* power
- Next 100X in less power?

(The Power Wall in Perspective)



The Memory Wall

The Power Wall

N. Jouppi, HP, The Future Evolution of High-Performance Microprocessors, Stanford EE 380

Towards the Next 100X: **The Complexity Wall**

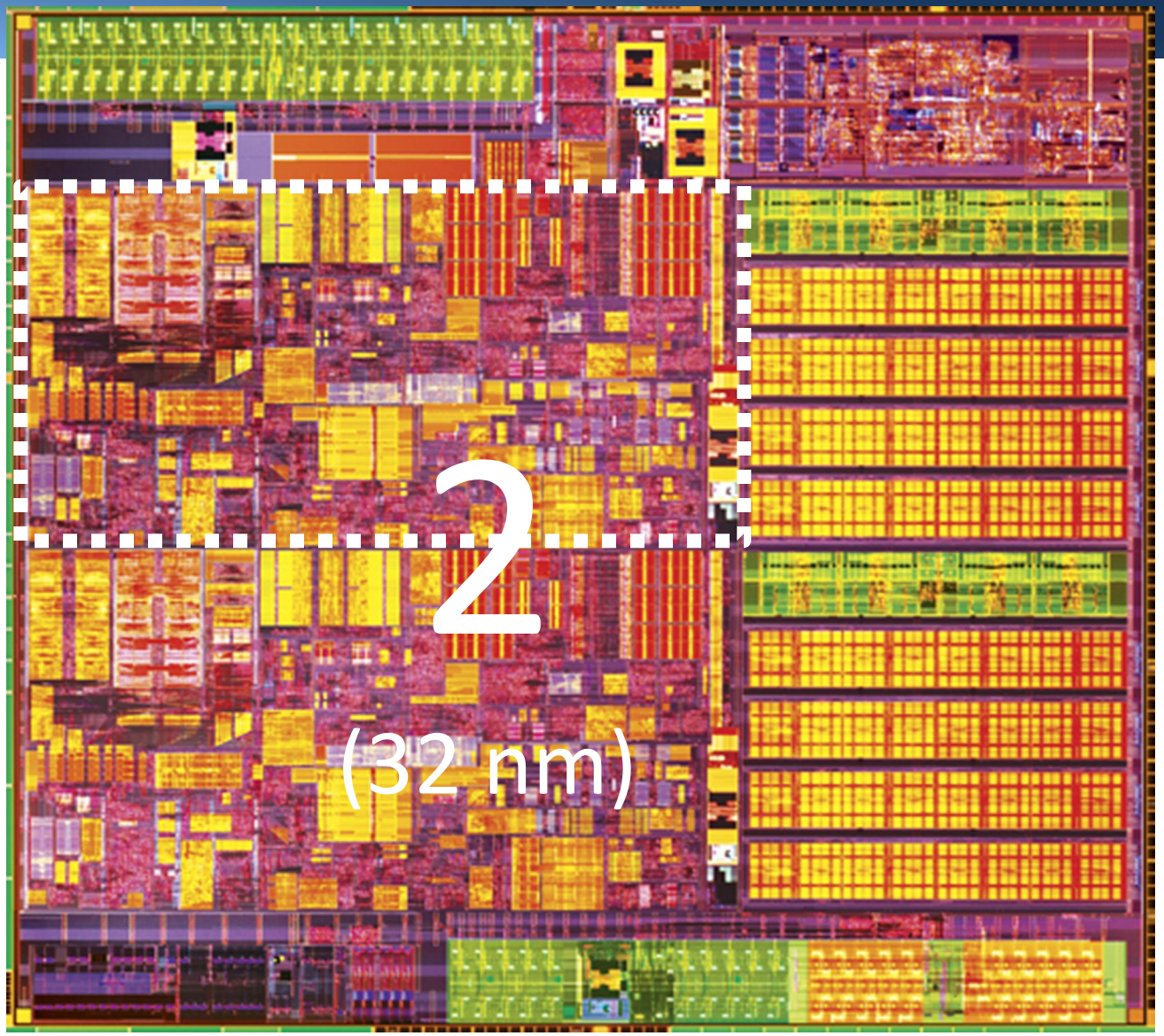
- Diminishing returns in more ILP
- Soaring design and verification time and cost
- *Only a small fraction of switching gates contribute to an answer*
- **Let's go back to simpler architectures**

Towards the Next 100X:
Explicit Parallelism – Multi-Core

- *"What else can we do with billions of transistors?"*
- **Idea**: chip-multiprocessors
  - Tile the die with lots of CPU cores – 2X *cores*/**2y**
  - Simpler cores → even more cores – hundreds!
- Finesse power with lower voltage and freq, sleep
- Finesse memory wall with memory parallelism

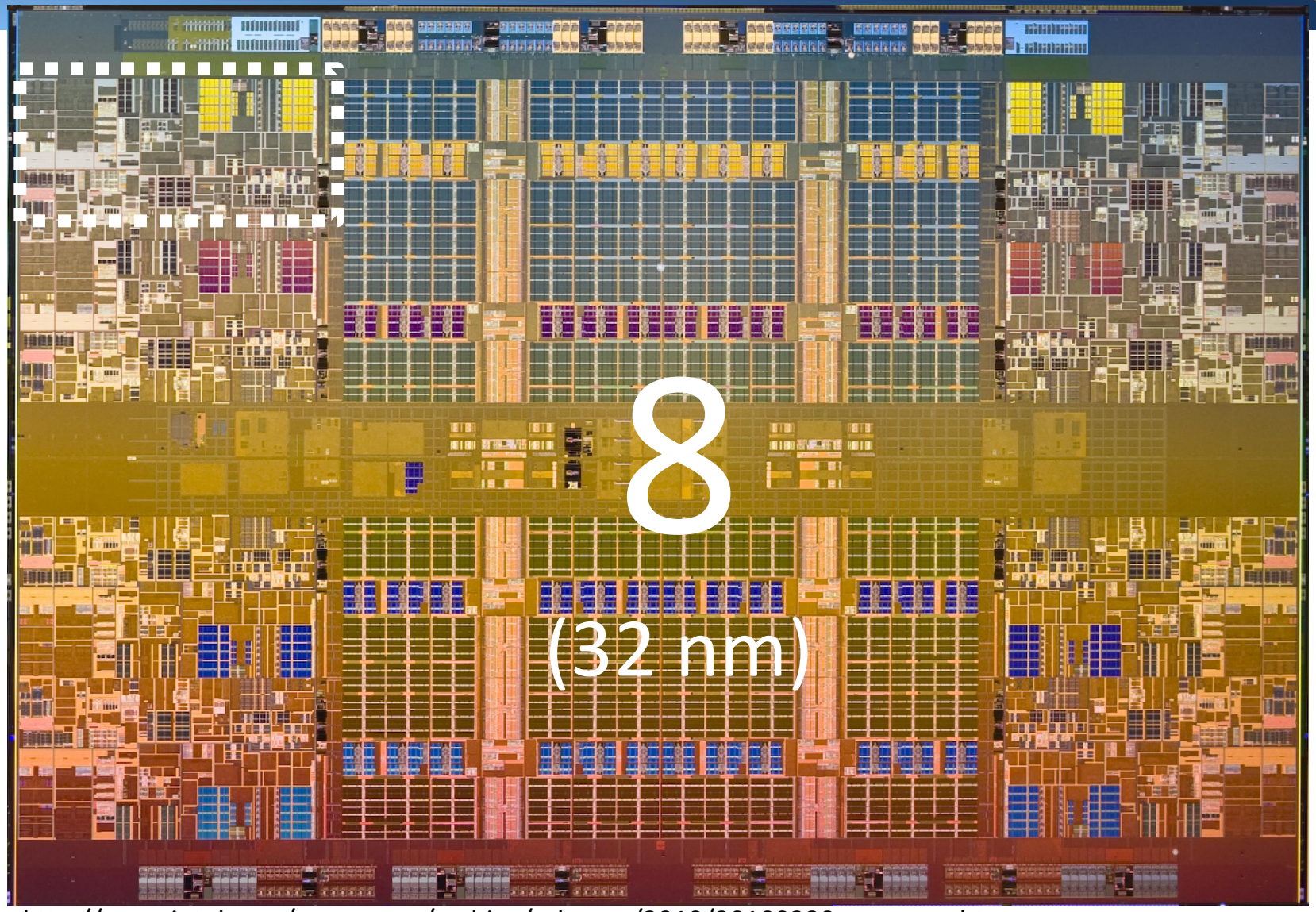
- *"We can't think of anything better.*
  Let those darn programmers deal with it!"

Chip-Multiprocessors: Laptop



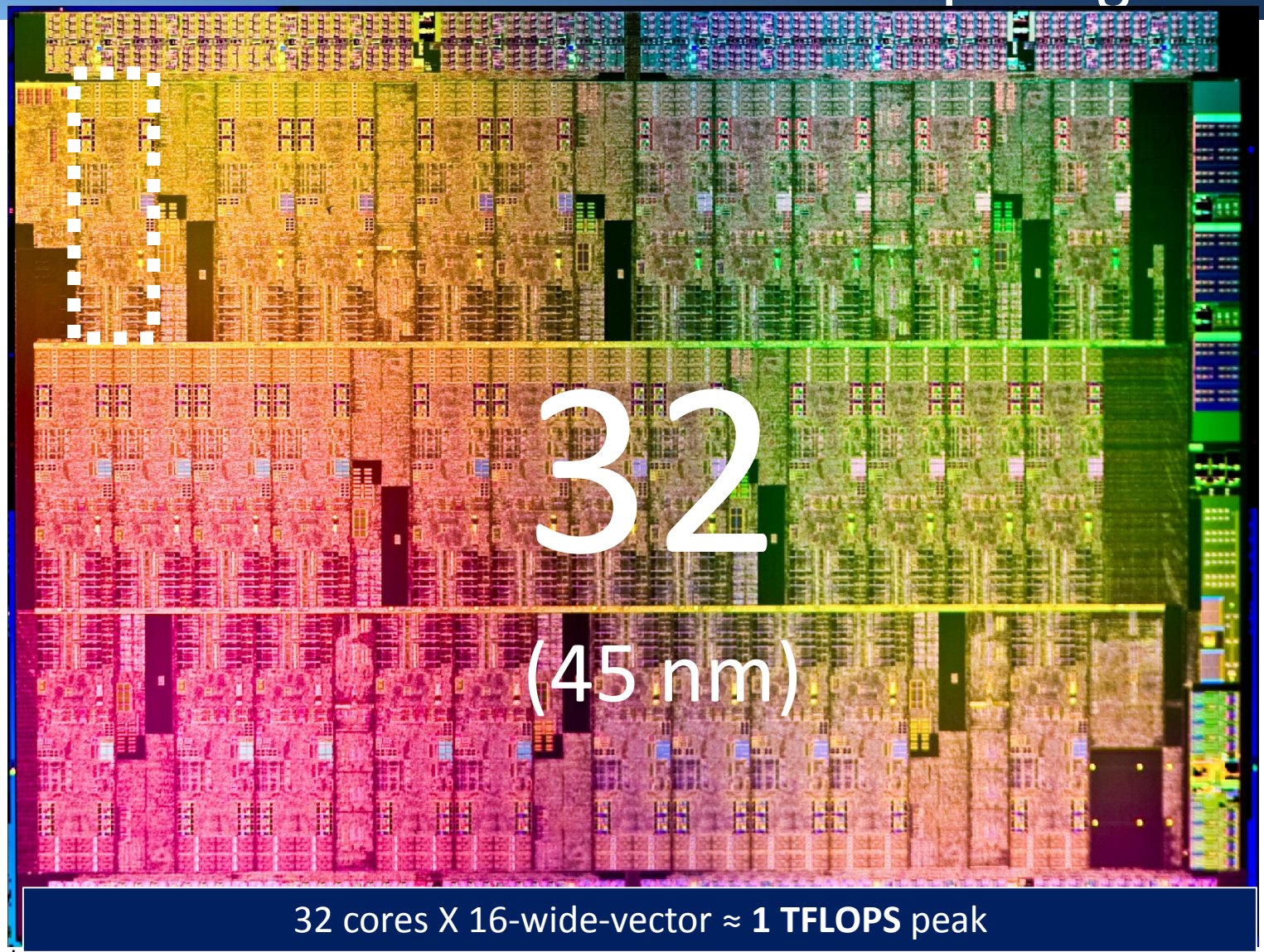
http://download.intel.com/pressroom/images/corefamily/Westmere%20Die%20Flat.jpg

Chip-Multiprocessors: Server



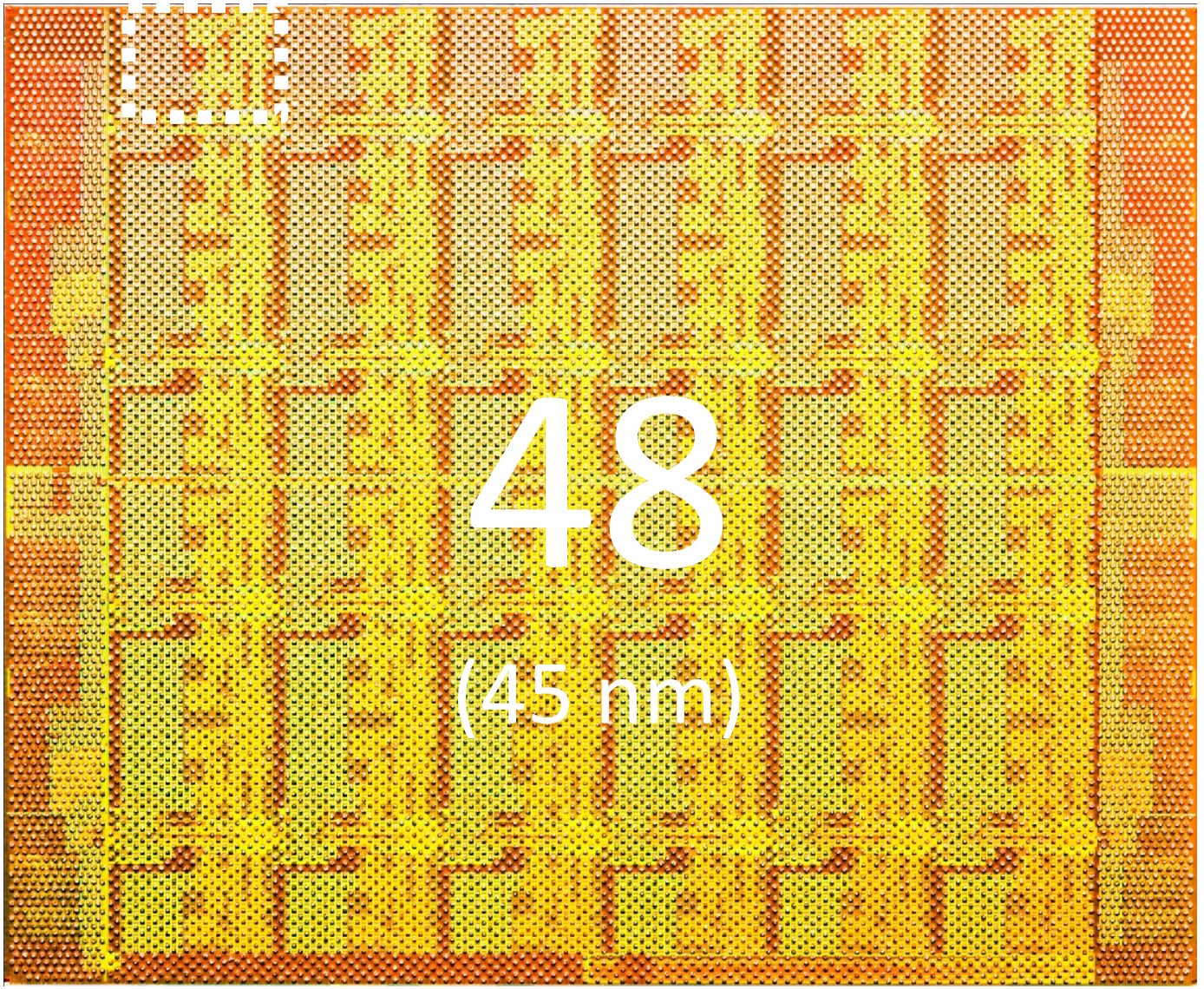
http://www.intel.com/pressroom/archive/releases/2010/20100330comp_sm.htm

Chip-Multiprocessors: High Performance Technical Computing



http://download.intel.com/pressroom/images/Aubrey_Isle_die.jpg

Chip-Multiprocessors: "Single-Chip Cloud" Datacenter on a Chip



http://download.intel.com/pressroom/images/rockcreek/scc-Chip.jpg

Scaling Up Memory Bandwidth:
New IC Packaging to "Feed the Beast"

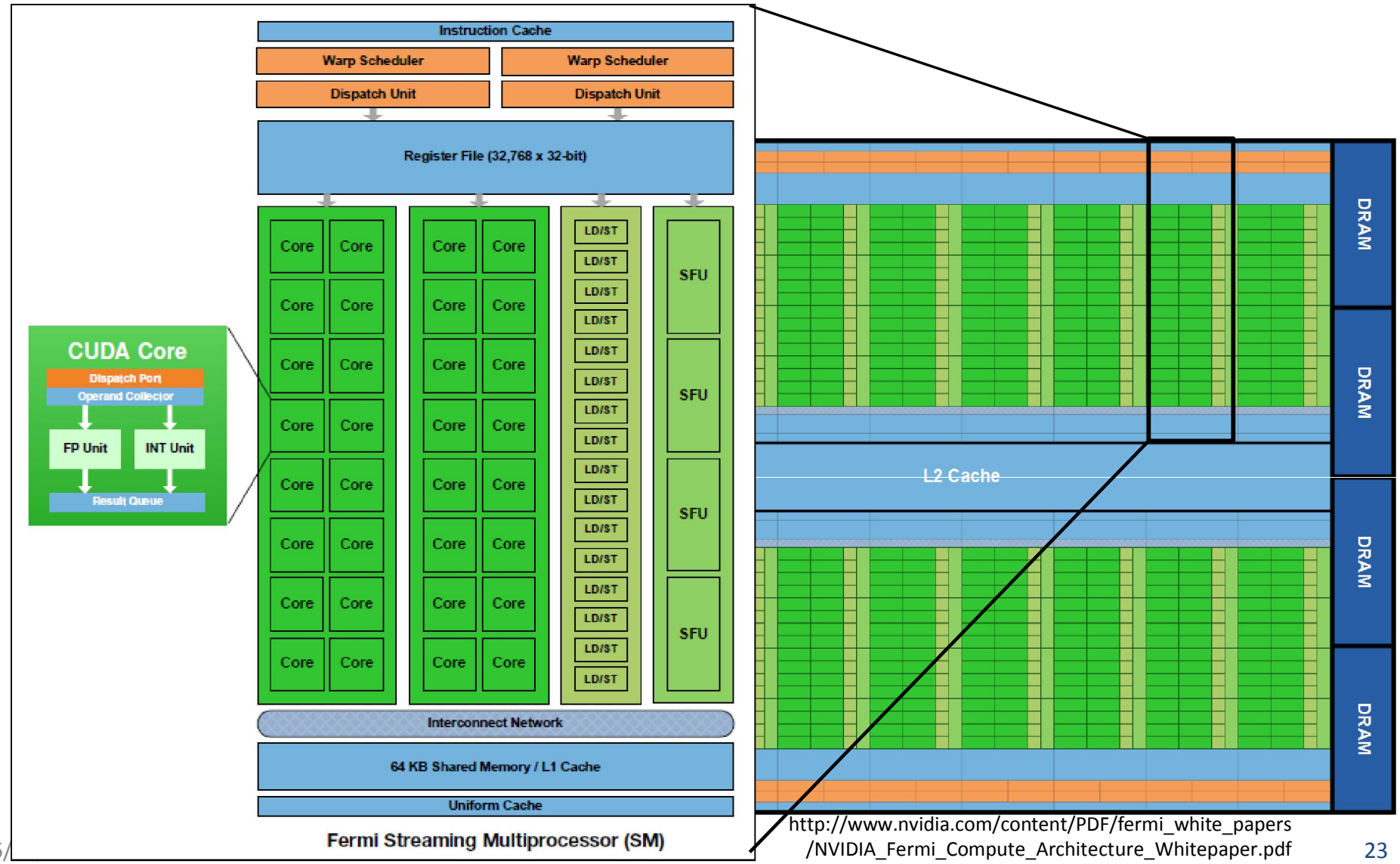
- Sustained teraflops need terabytes/s of data
  - Separate DRAM chips + every trick = not enough
- Stack DRAM onto CPU with 10,000 solder bumps
  - 1 TB/s at 1 GHz
  - Relatively power frugal
  - Mix CPU, DRAM, FLASH, optical dies in the stack

Scaling Up Bandwidth: Silicon Photonics

- Optical I/O should replace copper PCB buses
  - Low cost silicon scaling, integration, packaging
  - Many colors x 10s Gb/s/color ≈ Tb/s/channel

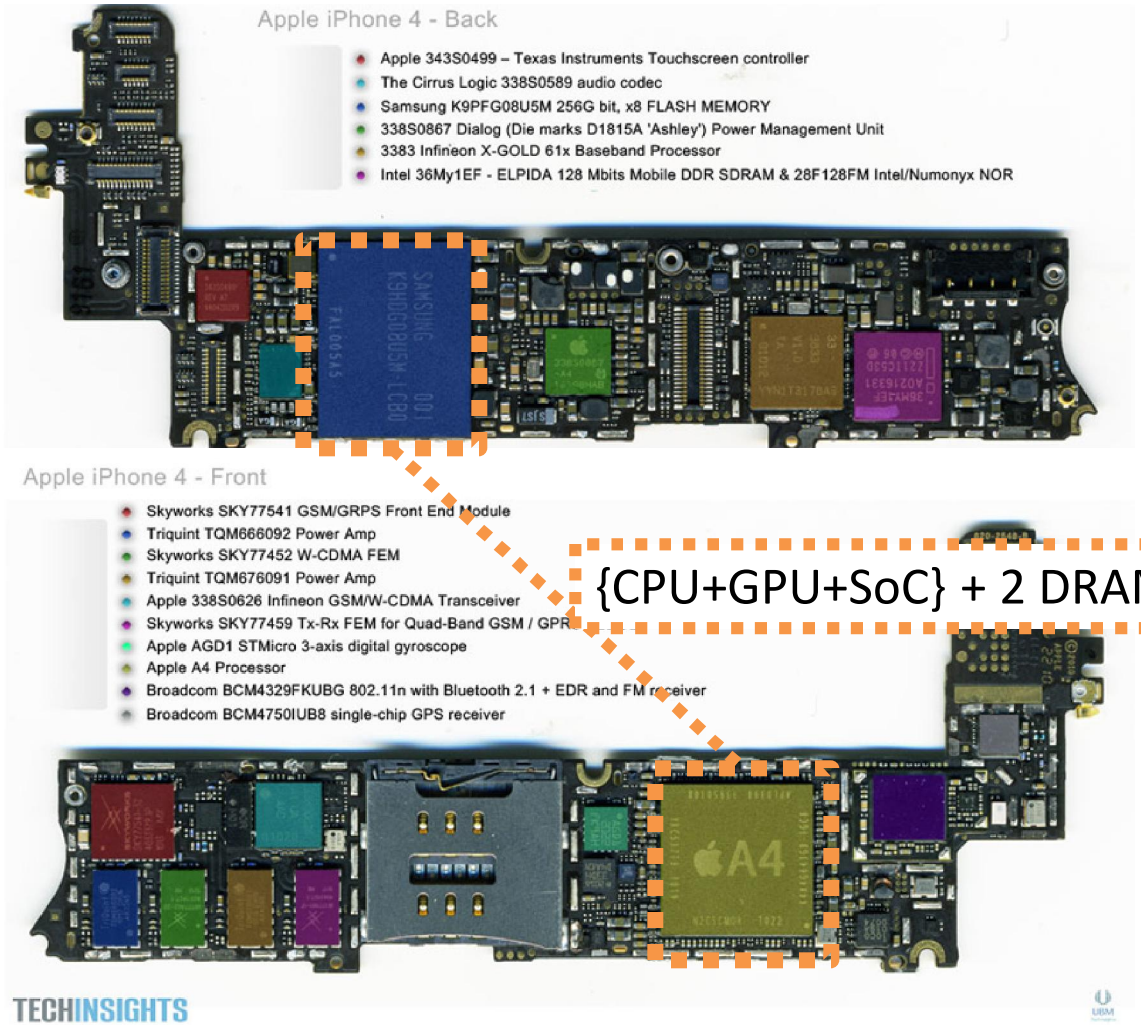
- Die stacking + photonics = adequate BW for ’10s
- 3D { processing + RAM + storage + IO } chips = “LEGO brick” computing nodes

Graphics Processor (GPU) Computing



http://www.nvidia.com/content/PDF/fermi_white_papers/NVIDIA_Fermi_Compute_Architecture_Whitepaper.pdf

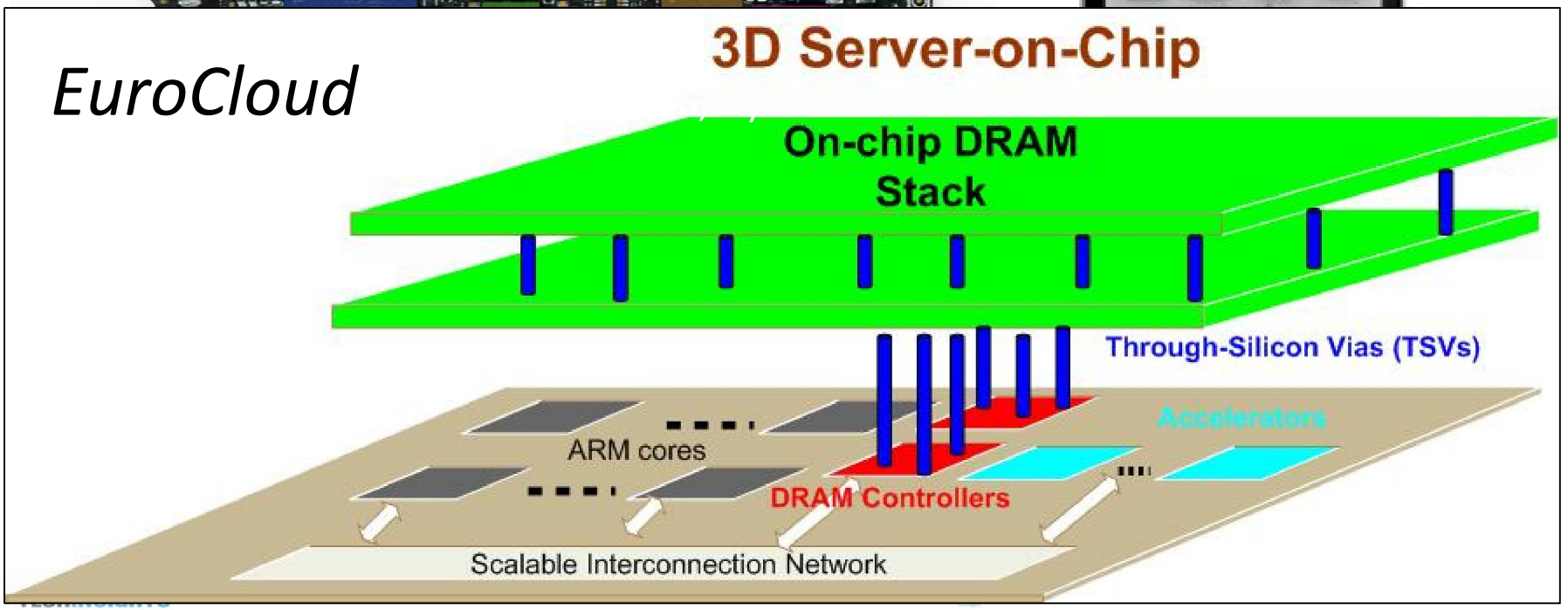
Putting It All Together – Of Phones and Data Centers



{CPU+GPU+SoC} + 2 DRAM + 8 FLASH = 11 dice!

Putting It All Together – Of Phones and Data Centers

EuroCloud



3. Mainstream Parallel Software

- Without parallel software, parallel machines are just fancy doorstops
- Some software is already parallel, already scales well
  - Databases; distributed tables; web servers; cloud services
  - Map-reduce parallel queries (Google, Bing)
  - Games: 3D graphics, game physics; immersive user interfaces
  - Technical computing on GPUs, clusters, and supercomputers
  - Computer vision; AV media; machine learning
  - *Independent, element-wise processing of huge data sets*
- Much useful software doesn't scale up on more cores

The Parallel Programming Model, Language, and Library Gap

- Most programmers “think serial”
- Old parallel models lead to “the pit of despair”
- Need a portfolio of new models and languages
  - Address diversity of problems, developers, legacies, cost/benefits, hardware
  - Some productive & safe, some explicit & risky
  - Package up expertise in reusable parallel libraries
- Good parallel code composability is imperative
- *Slow uptake by devs, and through software stack*

A Recipe for Scalable Parallel Programs

- Write programs with abundant *latent* parallelism
  - Invite parallel execution where needed, and safe
  - Over-decompose for future scaling
  - Use parallel libraries when possible
- System maps latent parallelism to available cores
- Employ *safe and scalable* models to avoid common pitfalls
  - Correctness bugs like *data races* and *deadlocks*
  - Performance bugs like *resource contention*

Amdahl's Law

- Alas the remaining serial parts of your code put an upper bound on scalability
  - Even with 1000 cores, a program that is 10% serial can only get a 10X speedup in the best case
- 100X speedups limited to problems that are totally parallel in nature
- Many algorithms, however expressed, retain a serial aspect
- Some software scales up, some can't

Prospects for Parallelization of Old Software?

- Legacy code “renovation” is a tough slog
  - Millions of lines of code
  - Serial assumptions baked into each interface layer
  - It’s hard to automate
  - Rewrite a few key performance bottlenecks
  - Or chase new rainbows: go parallel in new code
- Apps evolve as new parallel features grafted in

In Summary

- Transistor doublings continue, slowing down
- End of rapid scaling of old single threaded code
  - At least it's still scaling up – 10-15%/year is great
- 40%/year speedups for parallel software on parallel hardware through 2020
  - *Assuming* energy/computation falls 40%/year (??)

- Great disruptions bring great opportunities

One More Thing…

- At 14 nm, 10 nm, etc. *simple* systems-on-chips can be *tiny*, energy frugal, effectively free
  - Distributed smart objects, the internet of things
  - Enabling & demanding new kinds of software

So Let's Go Parallel

- Parallel programming exercises, with birthdays
- How to efficiently use the ~80 "parallel processors" here to compute:
  1. Does anyone have a birthday this week?
  2. How many here have a birthday in July?
  3. What is the last birthday of the year?
  4. How many have the same b'day as someone else here?
  5. What is our total age?
  6. What is our median age?
- What's easy? What's hard?
- How well would your algorithms scale to 320 students?

Key Questions

- Will market demand for faster chips continue?
- What matters most and can it be made parallel?
- Will we care more about low power mobile devices or high performance plugged-in ones?
- Will another paradigm save Moore's Law?

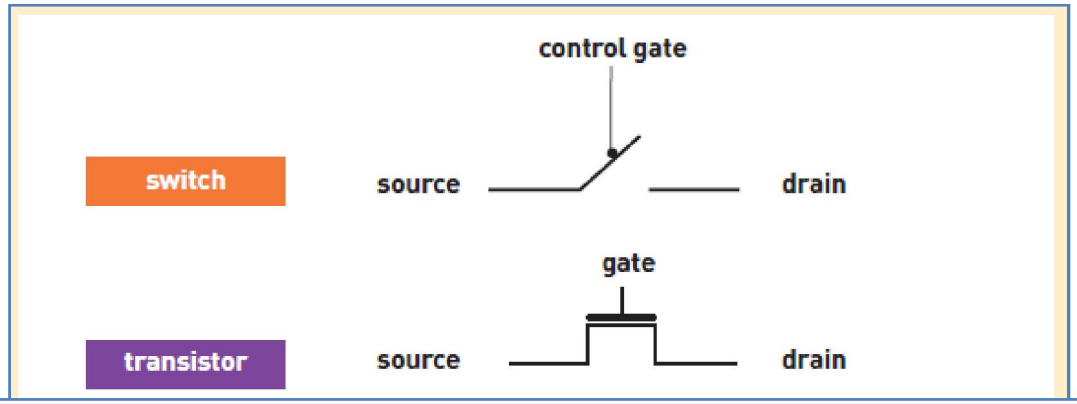
- Other comments, questions?

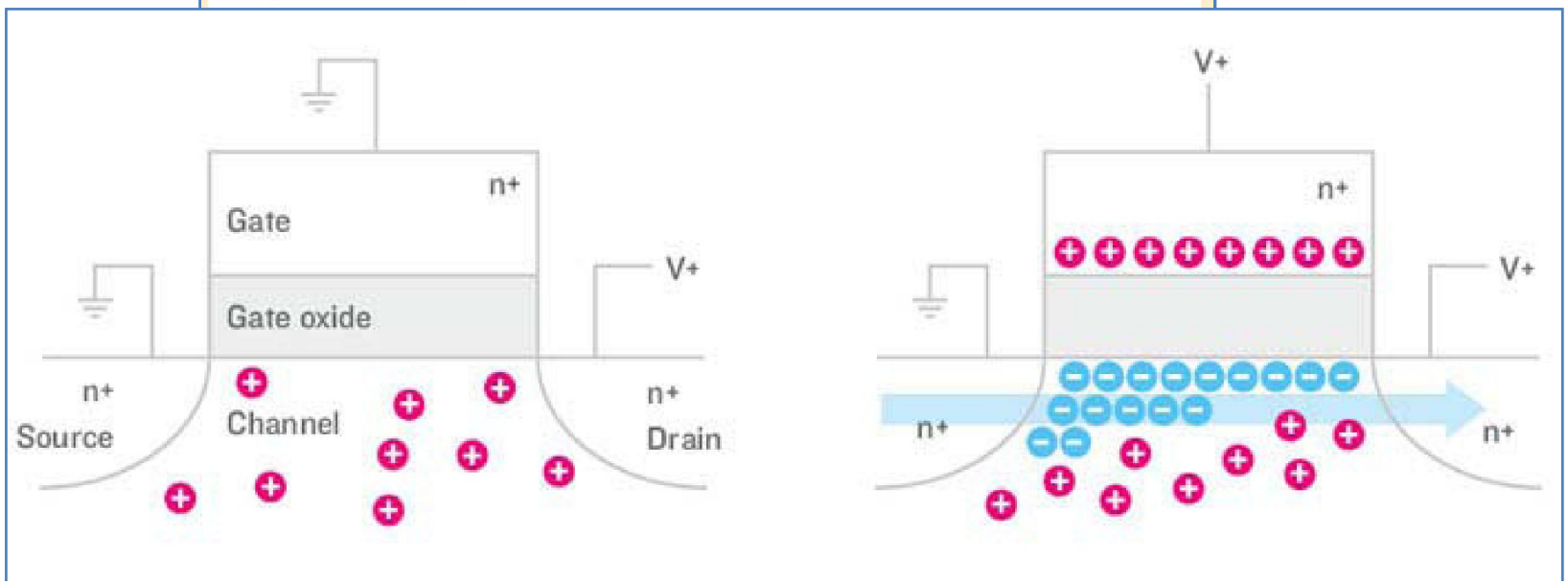
- *Thank you!*

Backup / Extra Material

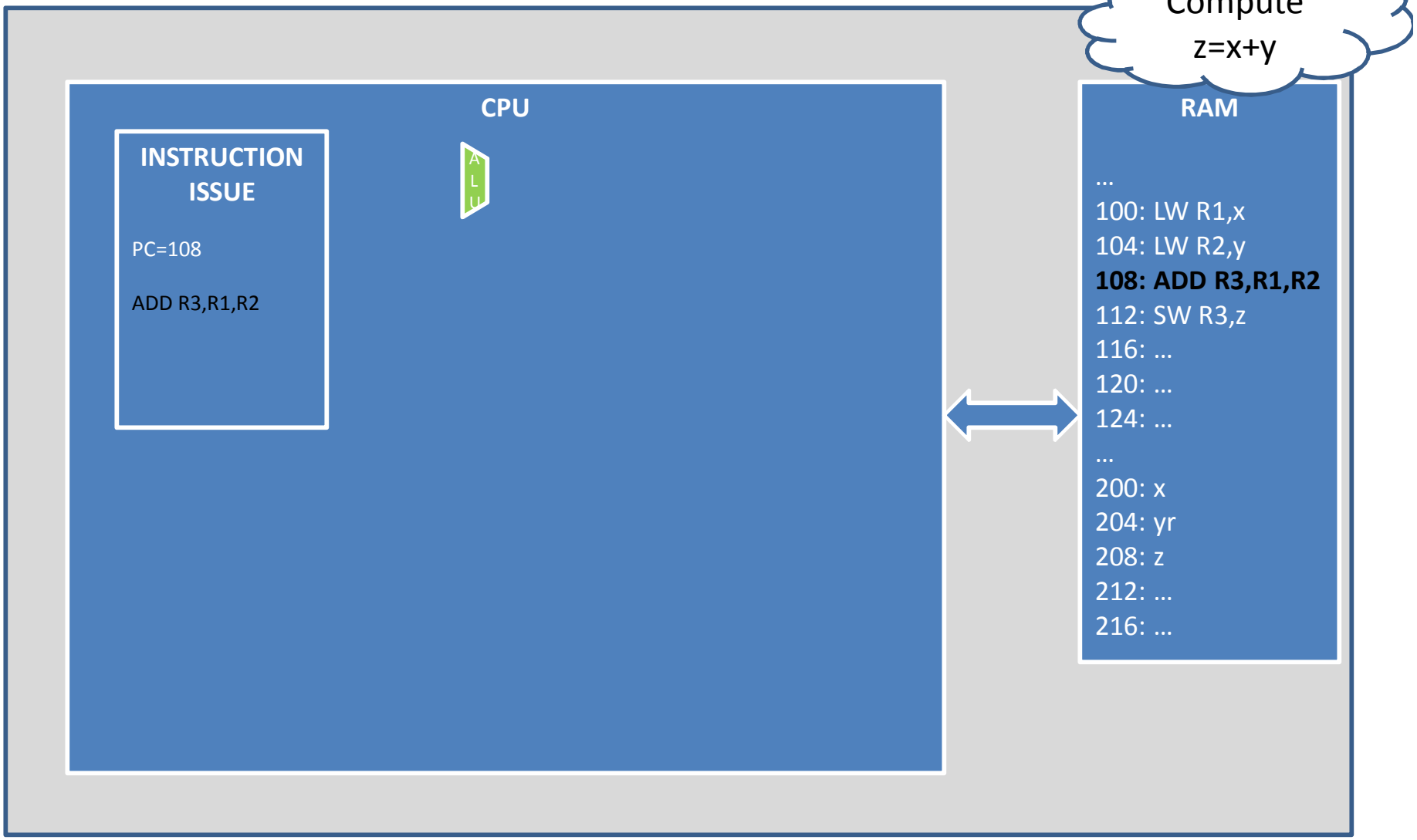
Moore's Law: The Marvelous MOSFET

- **M**etal gate, **O**xide insulator, **S**emiconductor channel **F**ield **E**ffect **T**ransistor

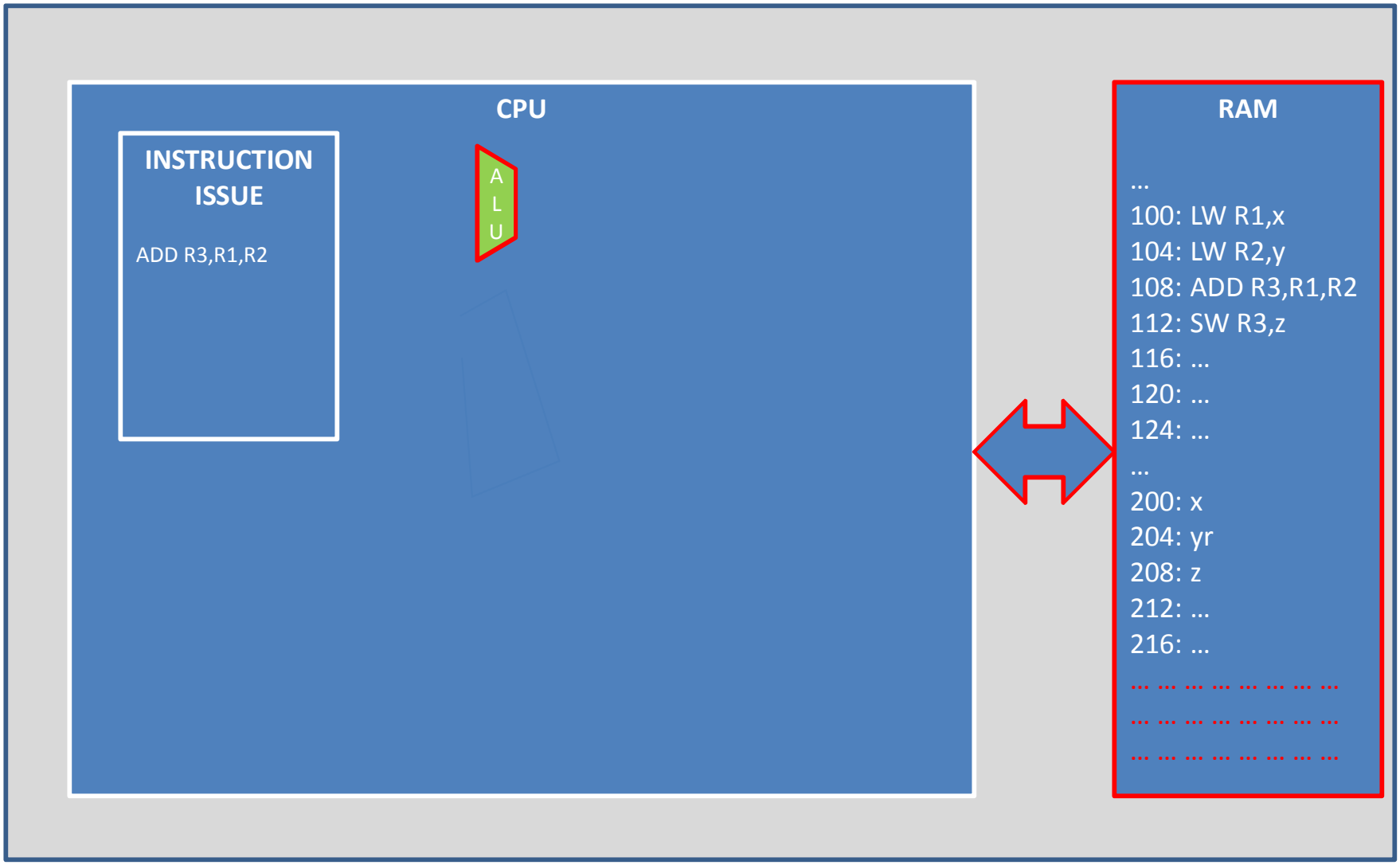




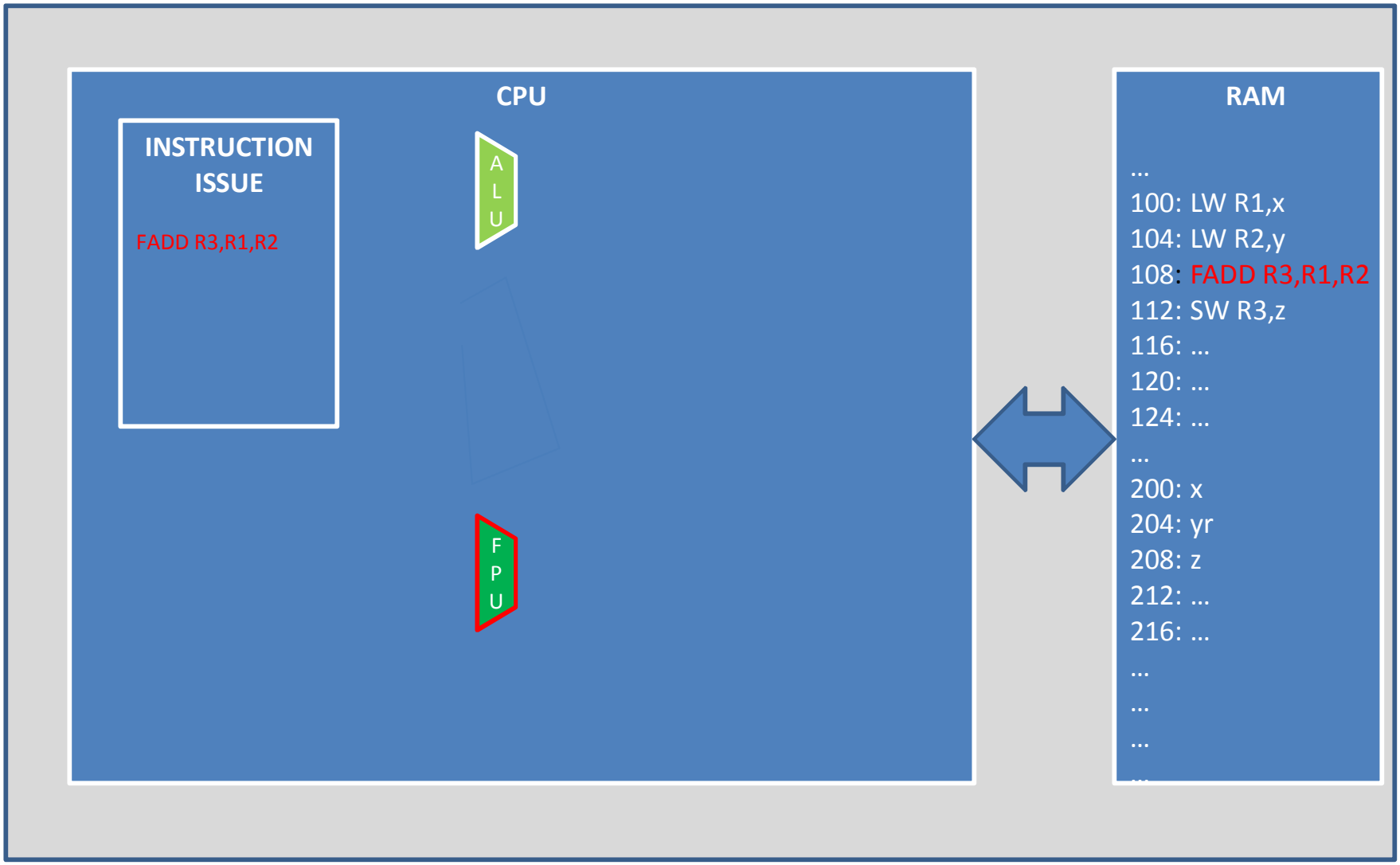
ILP: A Simple Computer



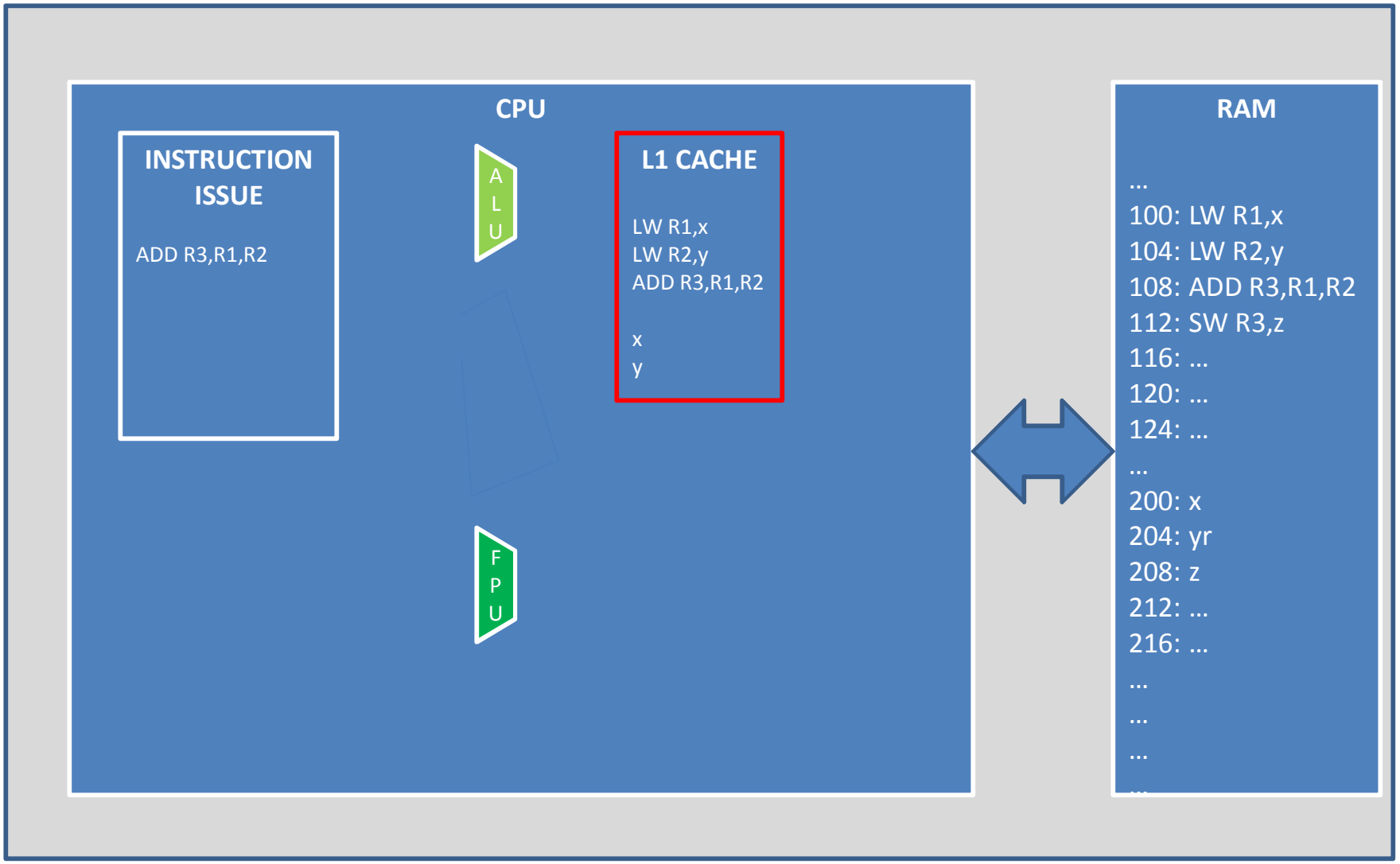
ILP: Widen Data and Addresses



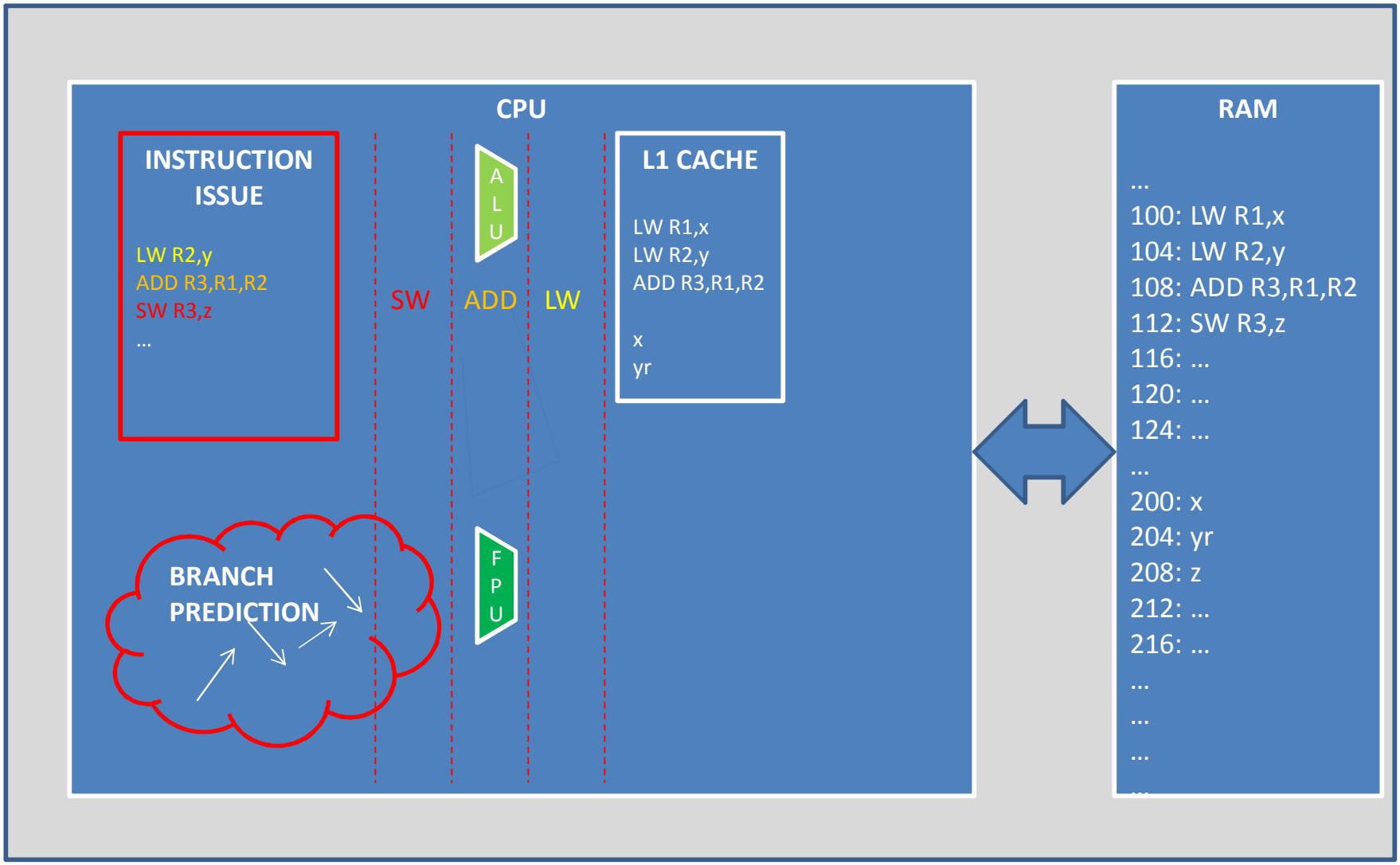
ILP: Add a Floating Point Unit



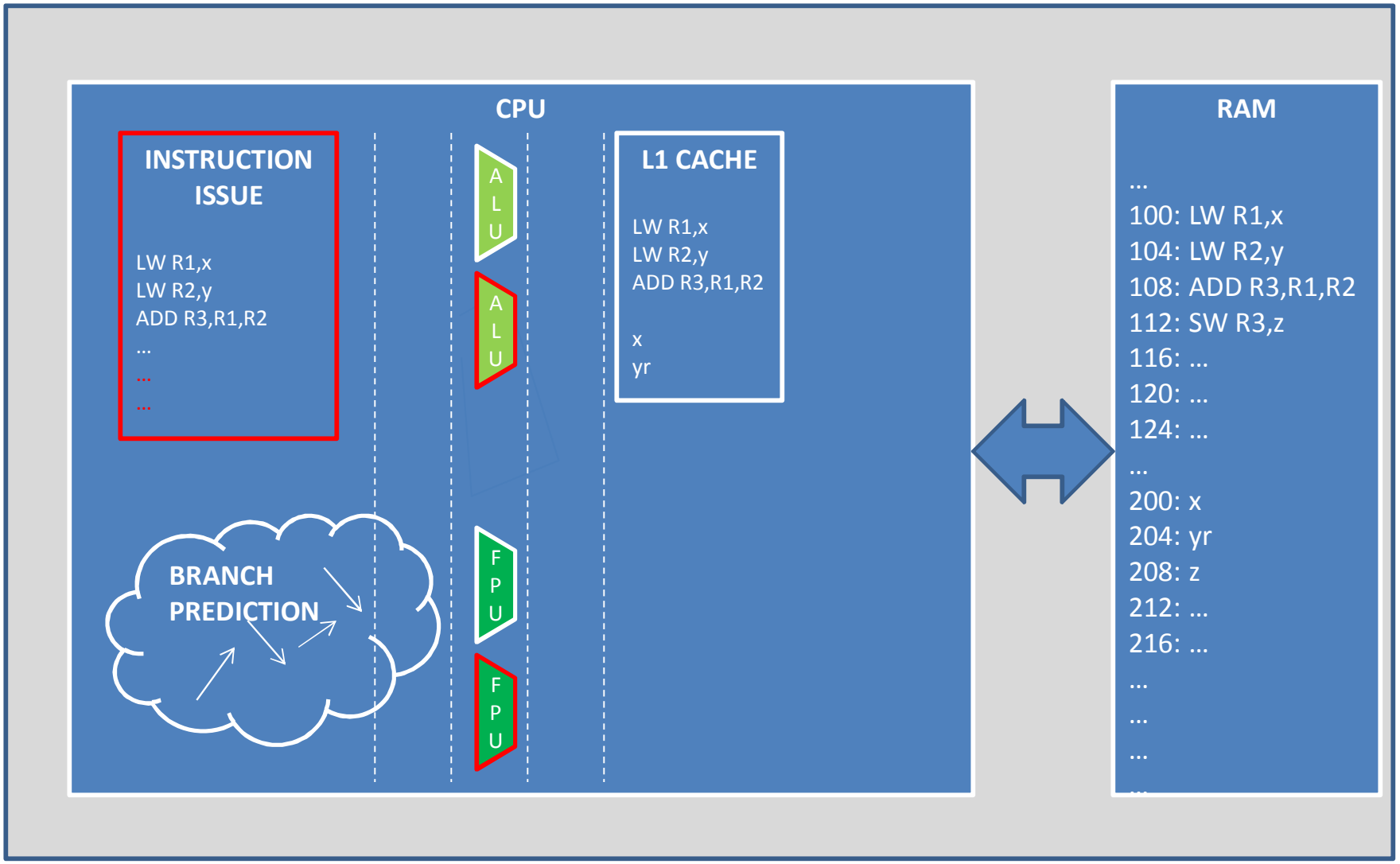
ILP: Add a Memory Cache



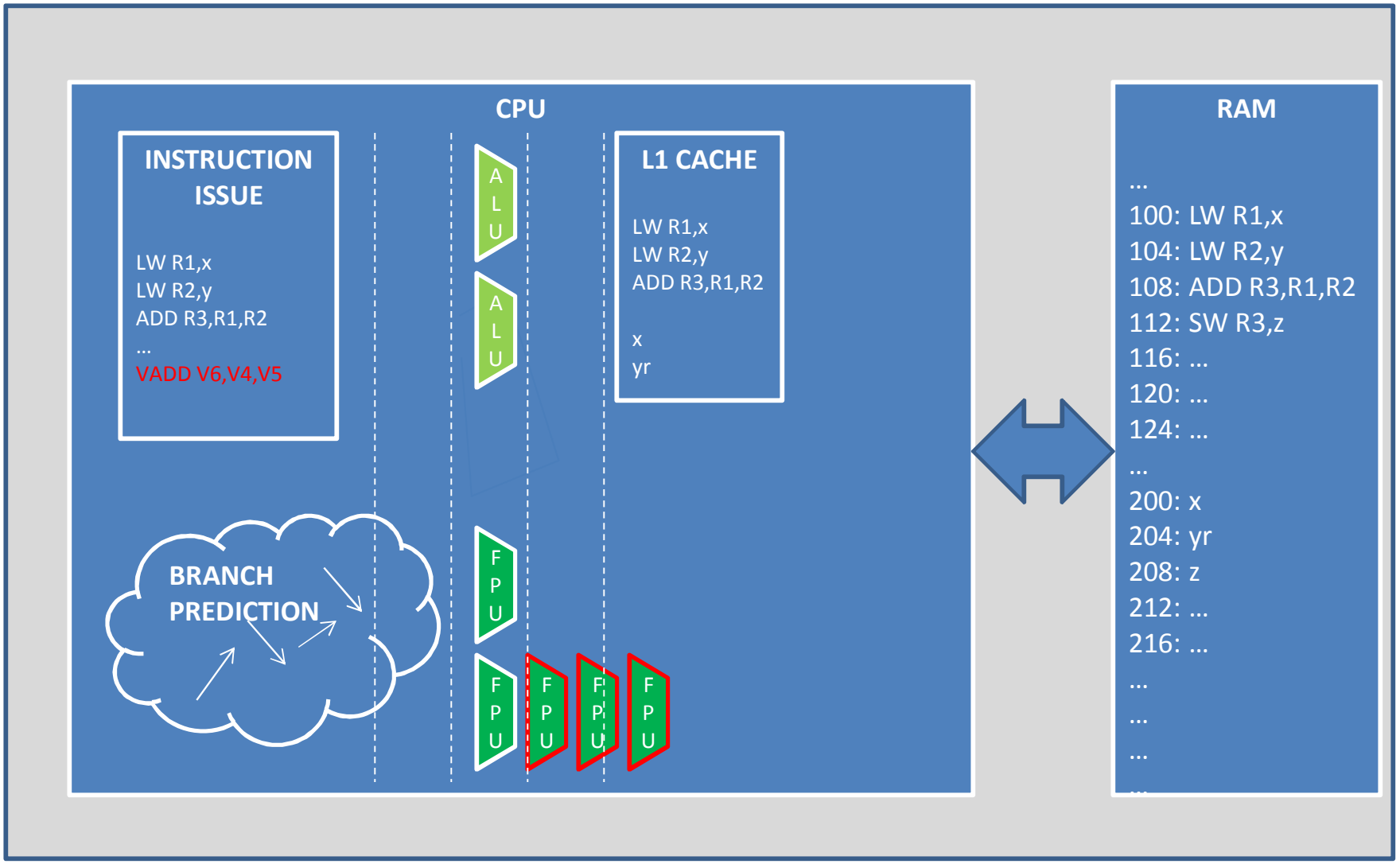
ILP: Add Pipelining



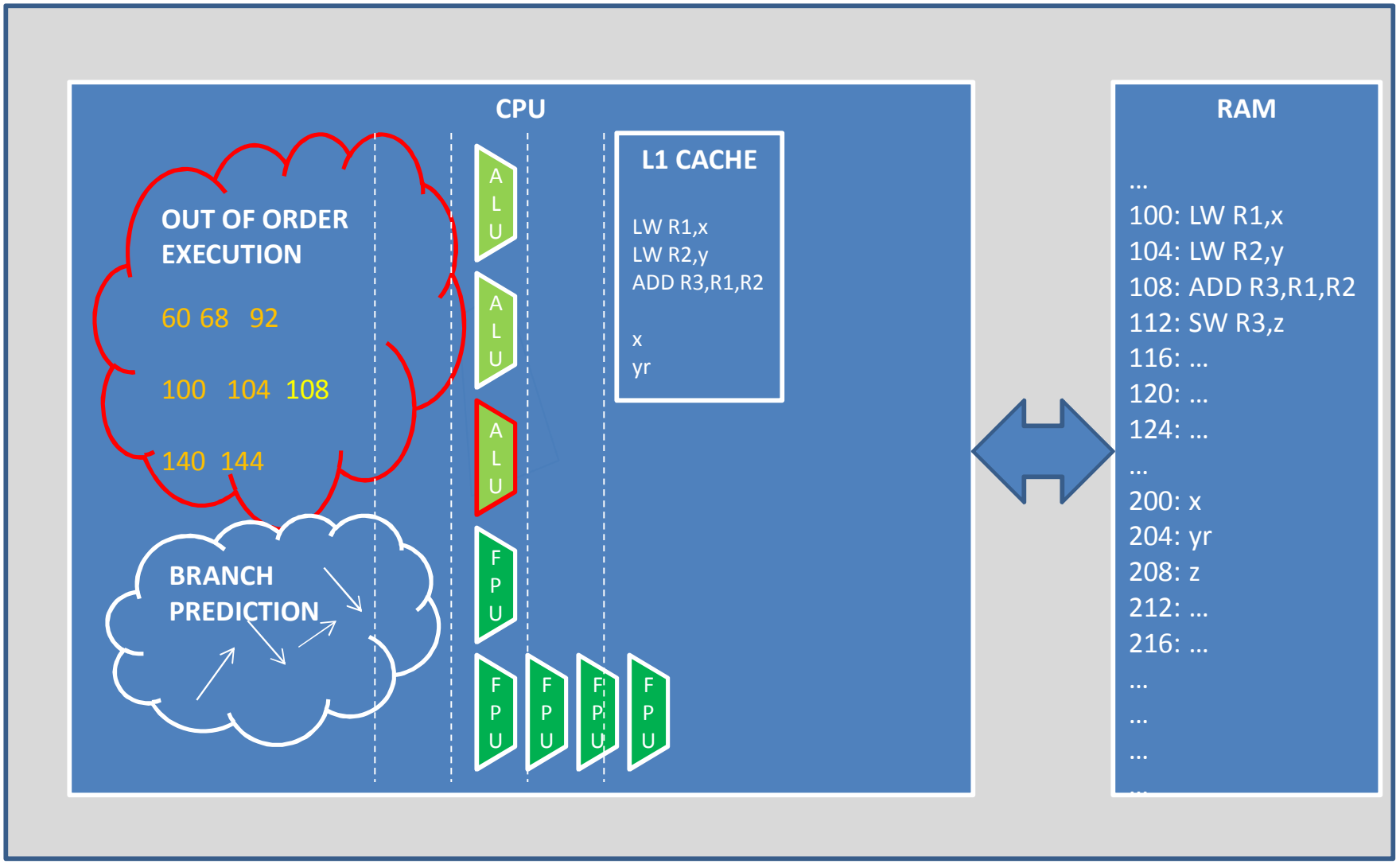
ILP: Issue Two Instructions Per Clock



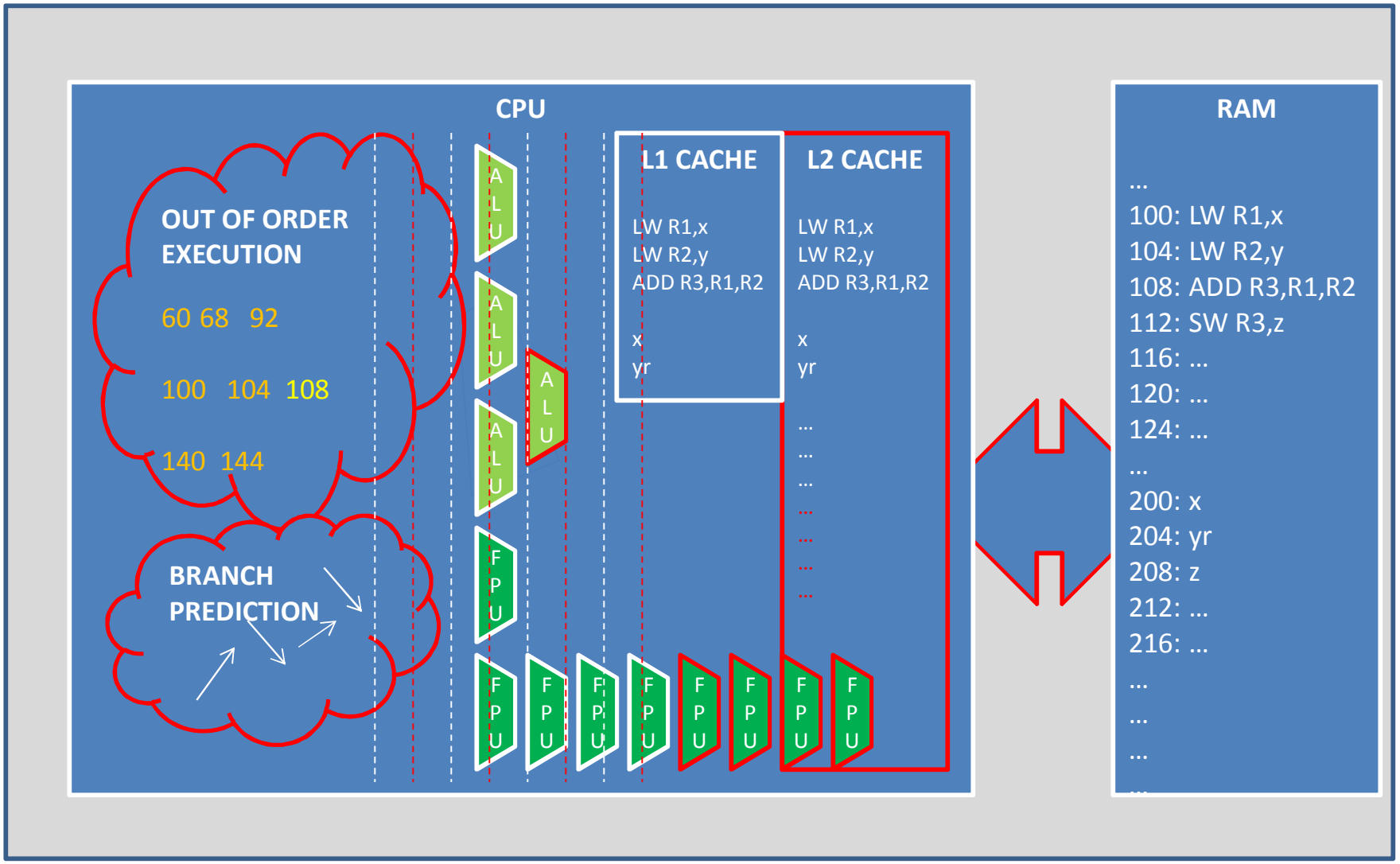
ILP: Add Vector Math Units



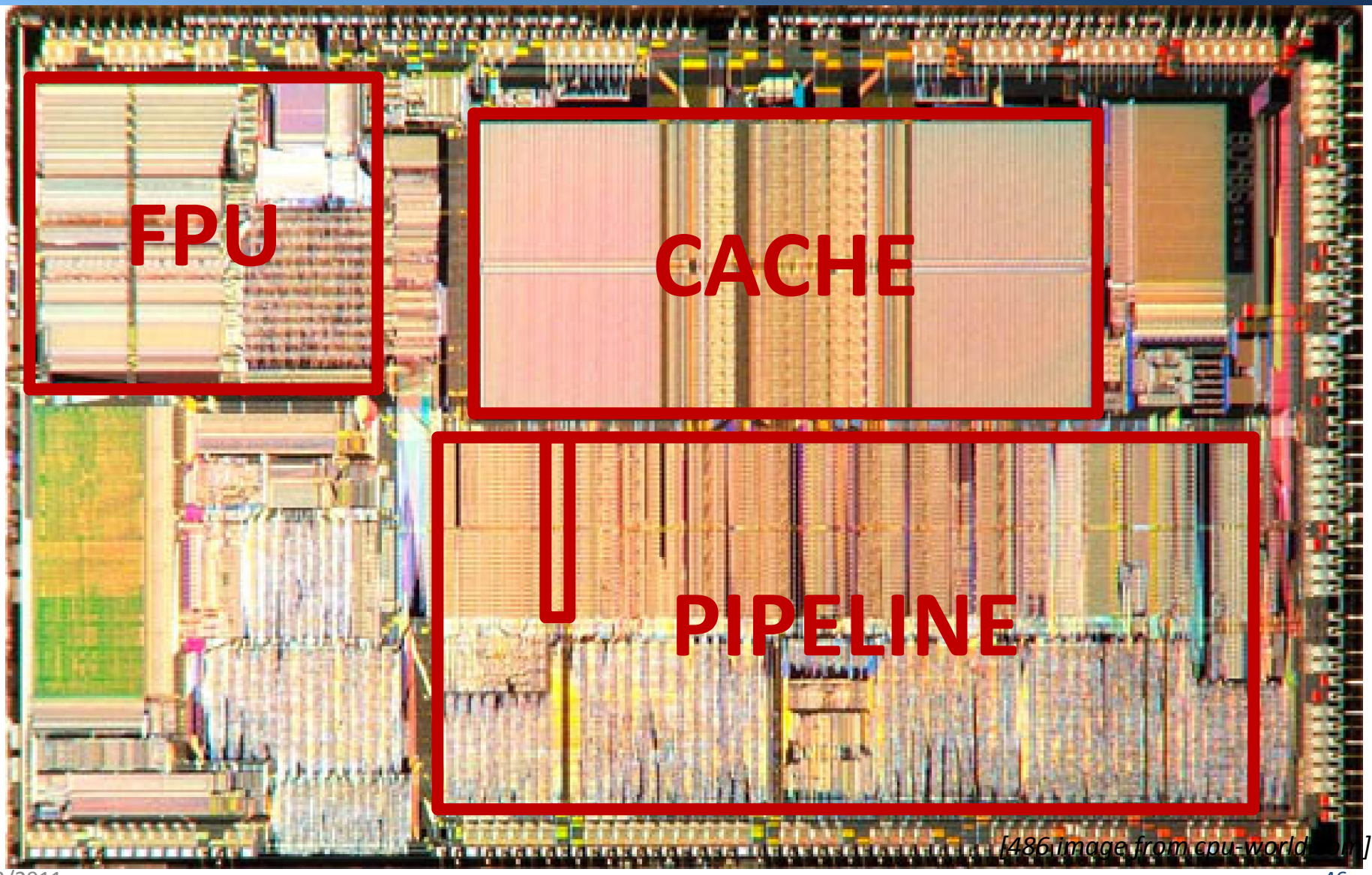
ILP: Add Out of Order Execution



ILP: Add More and More. When to Stop?



Example: Intel 486 (’89)



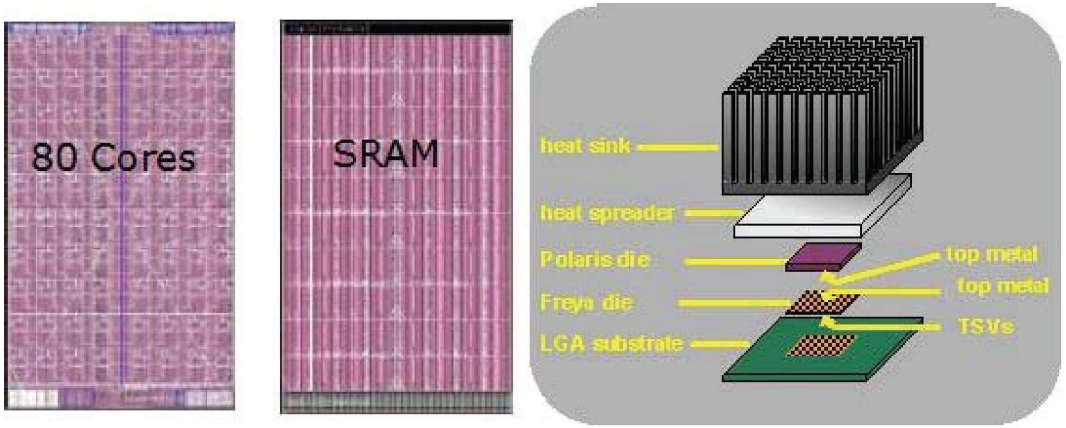
[486 image from cpu-world.com]

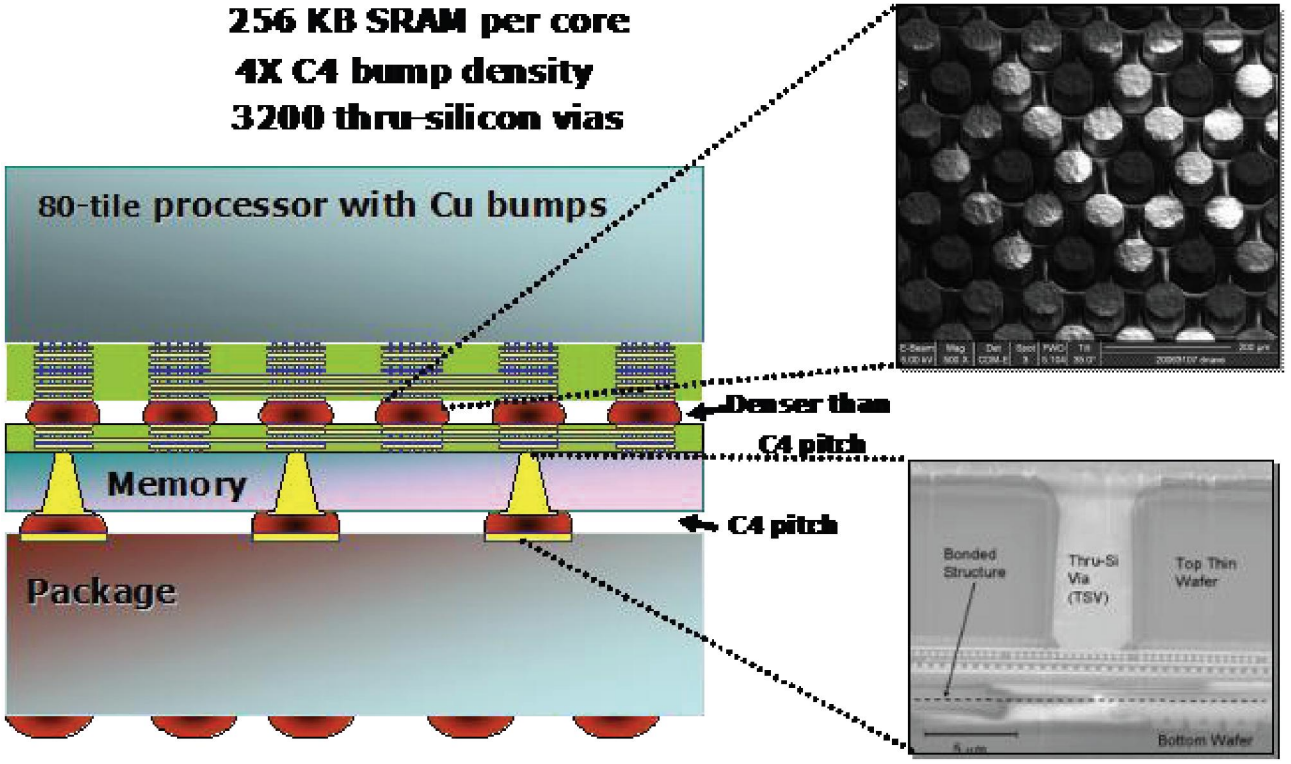
Graphics Processor (GPU) Computing

- Enormously parallel machines for **fast 3D graphics** …
  - 100s of simpler cores → teraflops at low energy/op
- … are also good for **data parallel computing**
  - Easier programming model and tools
  - Popular in technical computing
  - Optimizing for GPUs is still black magic
- Game enthusiasts fund next year's chip designs ☺

- Heterogeneity: put CPUs + GPU on same die

Scaling Up Bandwidth: Die Stacking





[from S. Borkar, Intel, VLSI-TSA, 2010]

Reconfigurable Computing (FPGAs)

- A sea of programmable gates and interconnect
  - Plus embedded RAMs, DSPs, 10 Gb/s links
  - It's SRAM – reconfigurable and scales with process
- Algorithms as custom hardware datapaths
- Enormous parallelism
  - e.g. 6 TB/s to RAM, 10^{15} bit-ops/s
- Field programmability comes at a cost
- Tools challenges

Shared Memory Considered Harmful

- All 3 examples scale well, but none were *robust*
- Except for "programmer discipline" two threads could read/write the same data at the same time
  - A *data race* – a pernicious, flakey bug
- This *shared memory* programming model is hard to use correctly, yet is the dominant paradigm
- Experts cope with it, but most developers will need new models that isolate their share of the data from other threads

Parallelism for Personal Computers Is Different Than Supercomputer Parallelism

- PC apps are **composed** of many libraries
  - Separately authored and versioned
  - Using diverse (parallel) languages, libraries, tools
- Binaries can live for decades
- Diversity of system topologies and capabilities, changing ms to ms, and year to year
- Bursty compute demands
- Diversity of developers …

Parallel HW and SW: Chicken and Egg

- Key market segments for parallel hardware
  - Immersive UI, games, tech computing, data centers
- Few mainstream PC apps showcase TFLOPS ↔ few commercial TFLOPS processors marketed
- "Relevant to my mom" test
- Yet cool new software always comes and brings new hardware to its knees
- "If we come, they can build it."