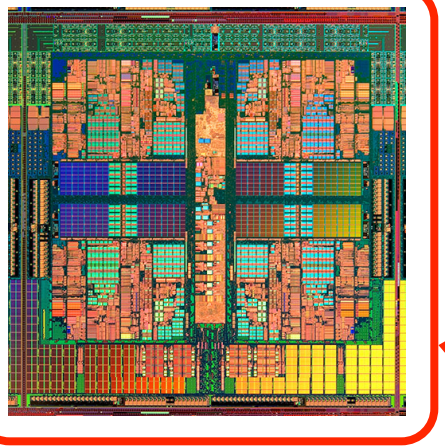
Lecture 28
Multicore, Multithread

Suggested reading:

(H&P Chapter 7.4)

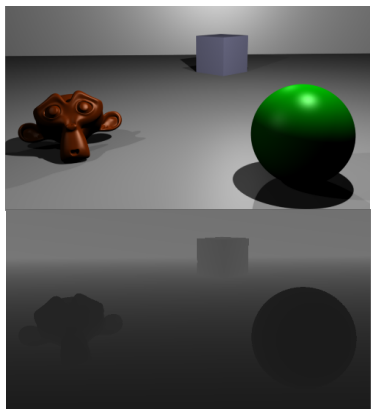
Multicore processors and programming



Goal: Explain and articulate why modern microprocessors now have more than one core and how software must adapt to accommodate the now prevalent multi-core approach to computing.

Processor components

Processor comparison



Writing more efficient code

The right HW for the right application

```
for i=0; i<5; i++ {
        a = (a*b) + c;
}
```

```
MULT r1,r2,r3    # r1 ← r2*r3
ADD r2,r1,r4     # r2 ← r1+r4
```

110011	000001	000010	000011
001110	000010	000001	000100

HLL code translation

Fundamental lesson(s)

- Some problems map well to parallel systems, others do not (and demand a fast, single thread).
- In this lecture, we will consider what classes of problems fall into each category

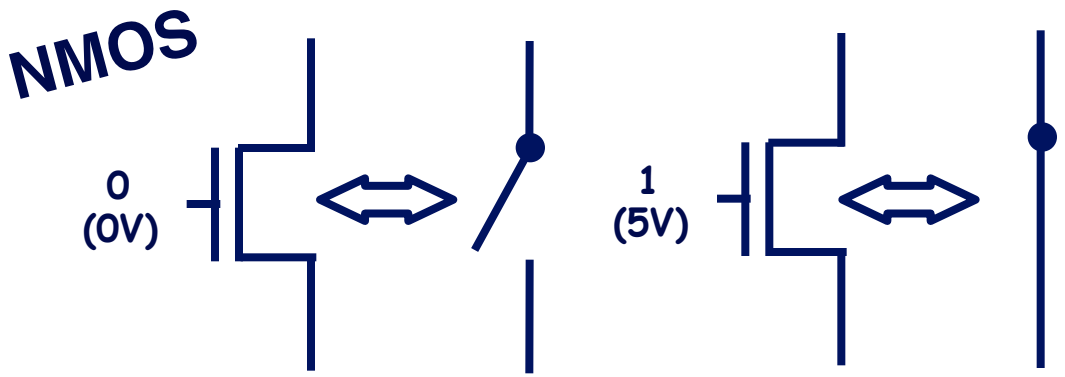
Why it’s important…

- **If you are writing software for a multi-core processor, and don't understand the implications / specifics of the underlying hardware, it's possible to write some very bad, ill-performing code.**

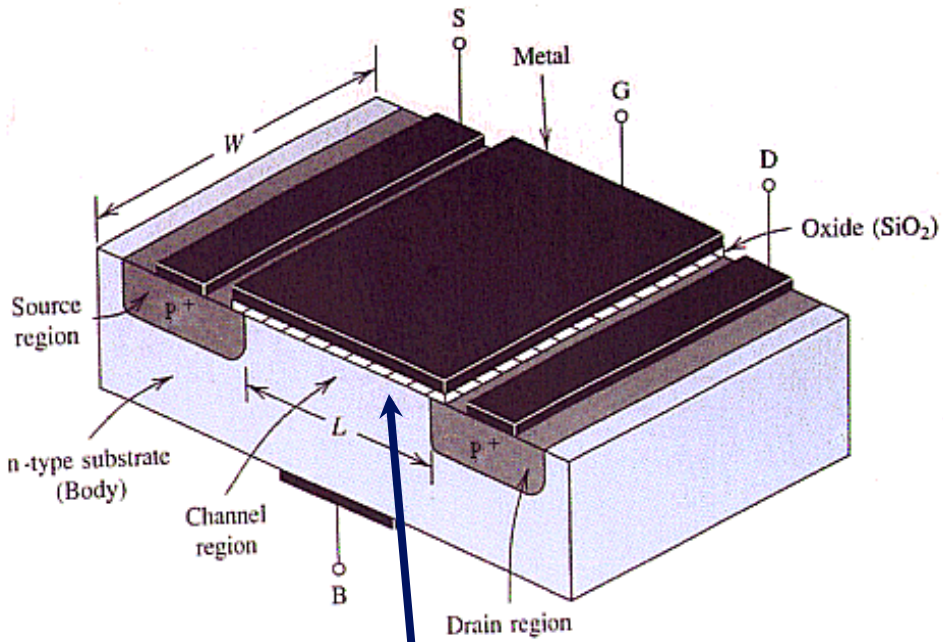
REMINDER: WHY MULTICORE?

Transistors used to manipulate/store 1s & 0s

Switch-level representation



Cross-sectional view



(can act as a capacitor storing charge)

Using above diagrams as context, note that if we (i) apply a suitable voltage to the gate & (ii) then apply a suitable voltage between source and drain, current will flow.

Moore's Law

- "Cramming more components onto integrated circuits."

  - G.E. Moore, Electronics 1965

  - Observation: DRAM transistor density doubles annually
    - Became known as "Moore's Law"
    - Actually, a bit off:
      - Density doubles every 18 months (now more like 24)
      - (in 1965 they only had 4 data points!)
  - Corollaries:
    - Cost per transistor halves annually (18 months)
    - Power per transistor decreases with scaling
    - Speed increases with scaling
      - Of course, it depends on how small you try to make things
        - » (I.e. no exponential lasts forever)

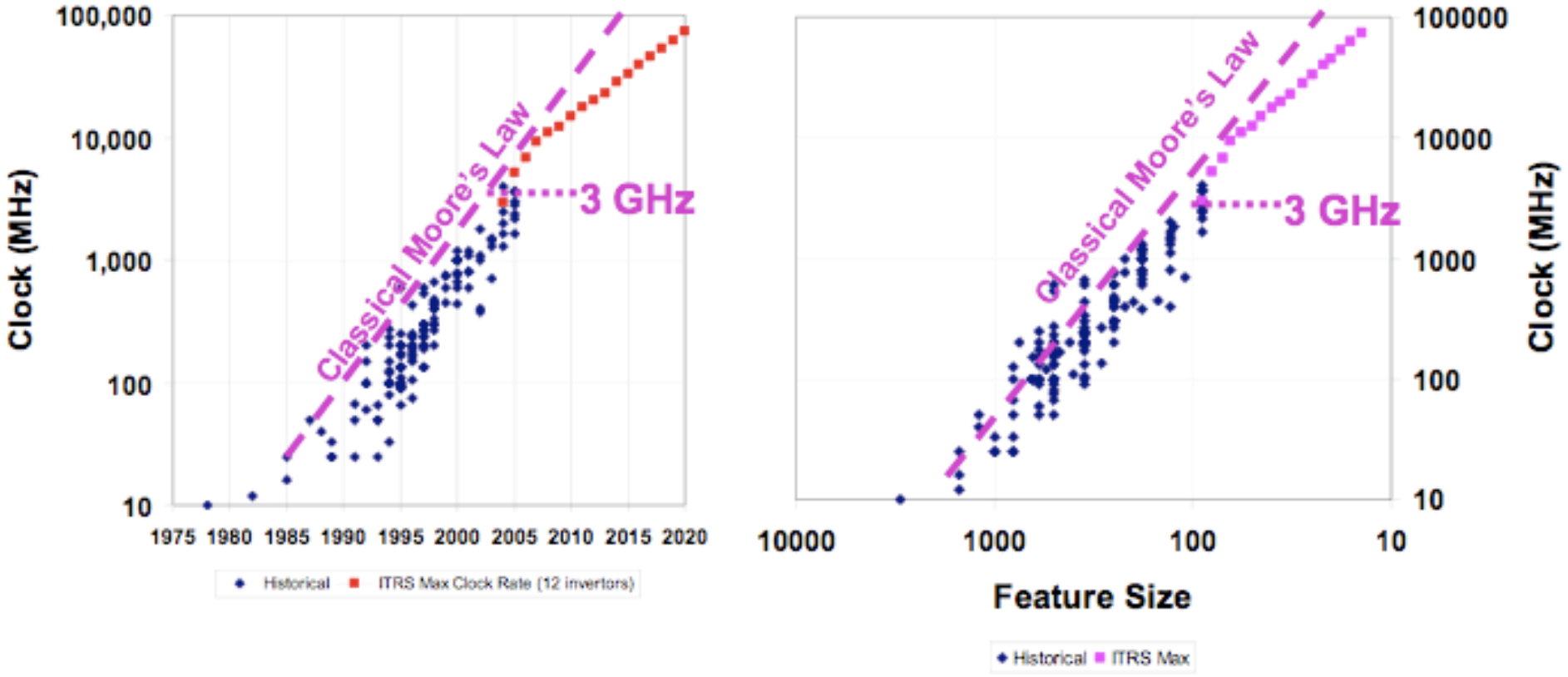
Remember these!

Previous Industry Projections

YEAR	2004	2007	2010	2013	2016
TECHNOLOGY	90 nm	65 nm	45 nm	32 nm	22 nm
CHIP SIZE	550 mm^2	550 mm^2	550 mm^2	550 mm^2	550 mm^2
NUMBER OF TRANSISTORS (LOGIC)	553 M	1 Billion	2 Billion	4.5 Billion	8.5 Billion
DRAM CAPACITY	1.0 Gbits	2.0 Gbits	4.3 Gbits	8.5 Gbits	35 Gbits
MAXIMUM CLOCK FREQUENCY	4.1 GHz	9.3 GHz	15 GHz	23 GHz	40 GHz
MINIMUM SUPPLY VOLTAGE	**0.9 V**	**0.8 V**	**0.7 V**	**0.6 V**	**0.5 V**
MAXIMUM POWER DISSIPATION	**150 W**	**190 W**	**200 W**	**200 W**	**200 W**
MAXIMUM NUMBER OF I/O PINS	3000	4000	4000	5300	7000

A funny thing happened on the way to 45 nm

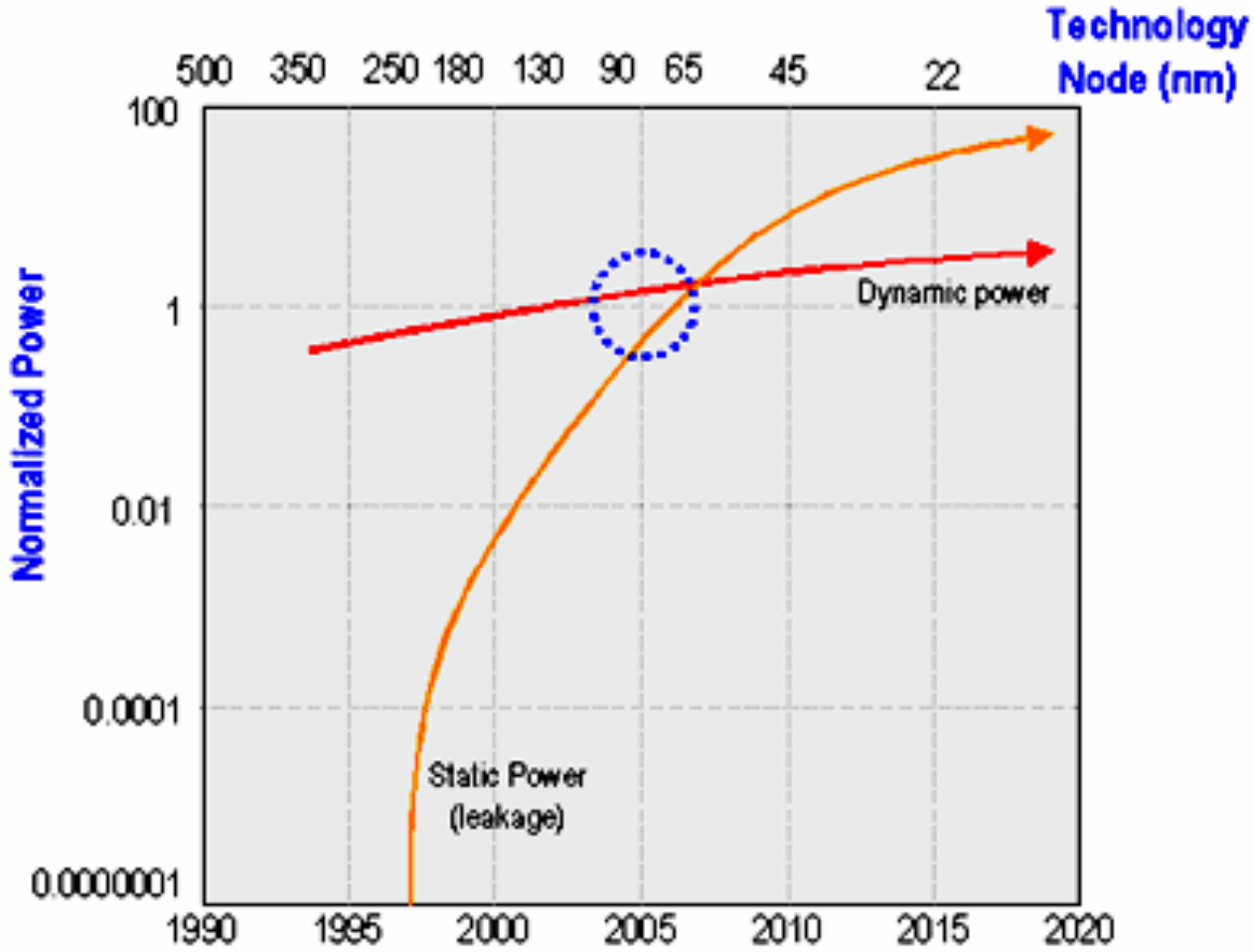
- Speed increases with scaling...



2005 projection was for 5.2 GHz - and we didn't make it in production. Further, we're still stuck at 3+ GHz in production.

A funny thing happened on the way to 45 nm

- Power decreases with scaling...

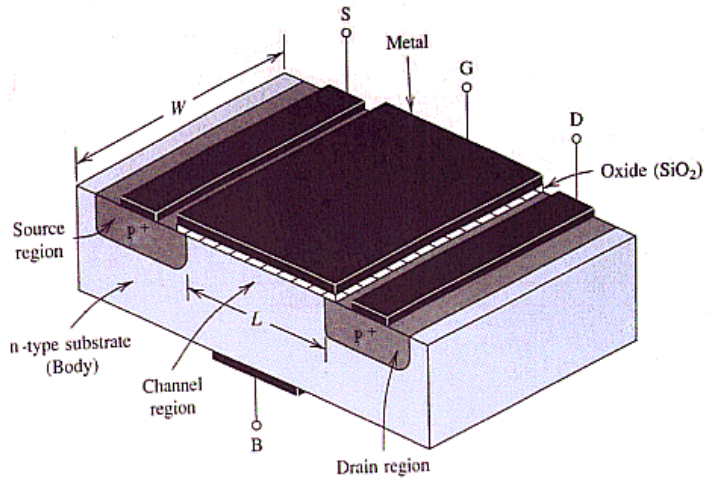


A bit on device performance...

- One way to think about switching time:
  - Charge is carried by electrons
  - Time for charge to cross channel = length/speed
    - = $L^2/(mV_{ds})$

Thus, to make a device faster, we want to either increase V_{ds} or decrease feature sizes (i.e. L)

- What about power (i.e. heat)?
  - Dynamic power is: $P_{dyn} = C_L V_{dd}^2 f_{0\text{-}1}$
    - $C_L = (e_{ox}WL)/d$
      - e_{ox} = dielectric, WL = parallel plate area, d = distance between gate and substrate



Summary of relationships

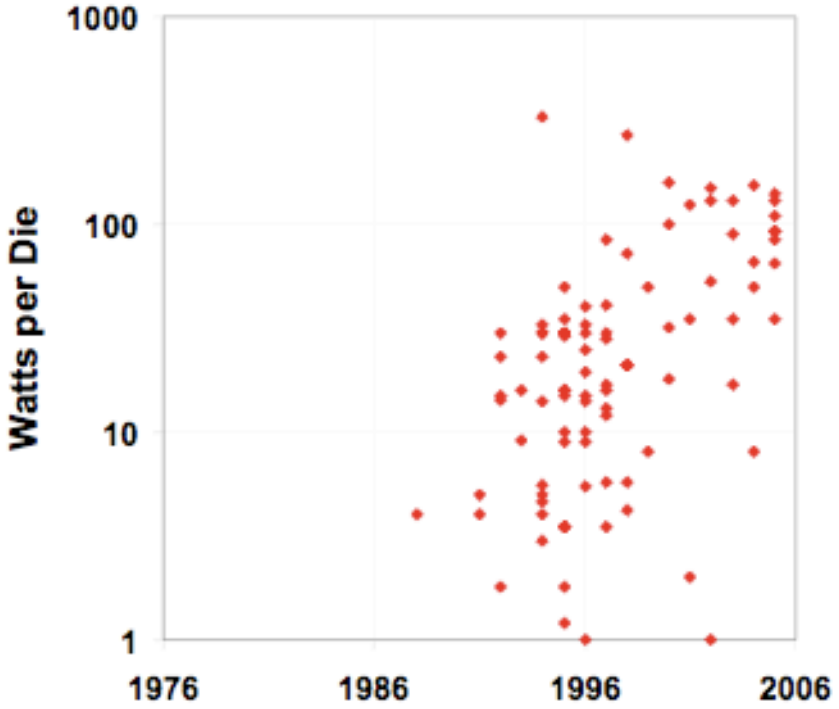
- (+) If V increases, speed (performance) increases
- (-) If V increases, power (heat) increases
- (+) If L decreases, speed (performance) increases
- (?) If L decreases, power (heat) does what?
  - P could improve because of lower C
  - P could increase because >> # of devices switch
  - P could increase because >> # of devices switch faster!

Need to carefully consider tradeoffs between speed and heat

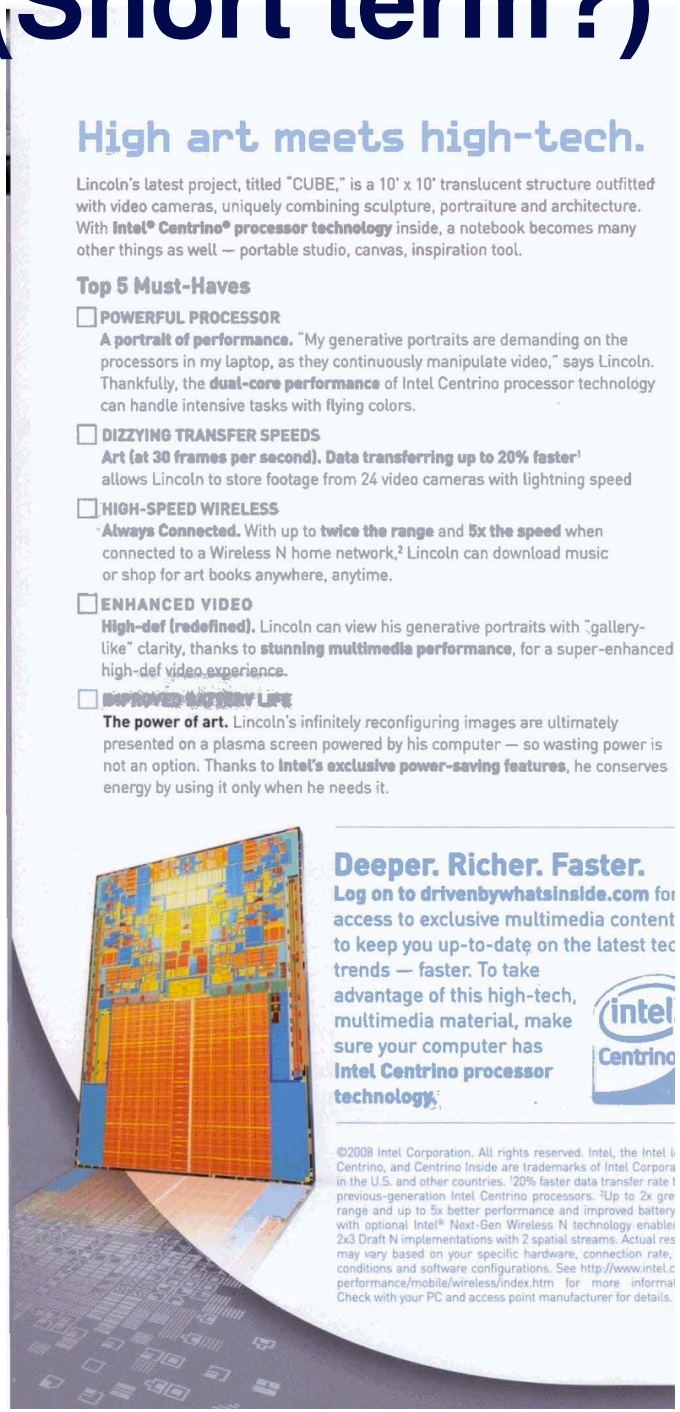
A funny thing happened on the way to 45 nm

- Speed increases with scaling...
- Power decreases with scaling...

Why the clock flattening? POWER!!!!



(Short term?) Solution



High art meets high-tech.

Lincoln's latest project, titled "CUBE," is a 10' x 10' translucent structure outfitted with video cameras, uniquely combining sculpture, portraiture and architecture. With **Intel® Centrino® processor technology** inside, a notebook becomes many other things as well — portable studio, canvas, inspiration tool.

Top 5 Must-Haves

☐ **POWERFUL PROCESSOR**
A portrait of performance. "My generative portraits are demanding on the processors in my laptop, as they continuously manipulate video," says Lincoln. Thankfully, the **dual-core performance** of Intel Centrino processor technology can handle intensive tasks with flying colors.

☐ **DIZZYING TRANSFER SPEEDS**
Art (at 30 frames per second). Data transferring up to 20% faster[1] allows Lincoln to store footage from 24 video cameras with lightning speed

☐ **HIGH-SPEED WIRELESS**
Always Connected. With up to **twice the range** and **5x the speed** when connected to a Wireless N home network,[2] Lincoln can download music or shop for art books anywhere, anytime.

☐ **ENHANCED VIDEO**
High-def (redefined). Lincoln can view his generative portraits with "gallery-like" clarity, thanks to **stunning multimedia performance**, for a super-enhanced high-def video experience.

☐ **[illegible]**
The power of art. Lincoln's infinitely reconfiguring images are ultimately presented on a plasma screen powered by his computer — so wasting power is not an option. Thanks to **Intel's exclusive power-saving features**, he conserves energy by using it only when he needs it.

Deeper. Richer. Faster.

Log on to drivenbywhatsinside.com for access to exclusive multimedia content to keep you up-to-date on the latest tec trends — faster. To take advantage of this high-tech, multimedia material, make sure your computer has **Intel Centrino processor technology**.

intel Centrino

- **Processor complexity is good enough**
- **Transistor sizes can still scale**
- **Slow processors down to manage power**
- **Get performance from...**

Parallelism

Top 5 Must-Haves

☐ POWERFUL PROCESSOR
A portrait of performance. "My generative portraits are demanding on the processors in my laptop, as they continuously manipulate video," says Lincoln. Thankfully, the **dual-core performance** of Intel Centrino processor technology can handle intensive tasks with flying colors.

(i.e. 1 processor, 1 ns clock cycle
vs.
2 processors, 2 ns clock cycle)

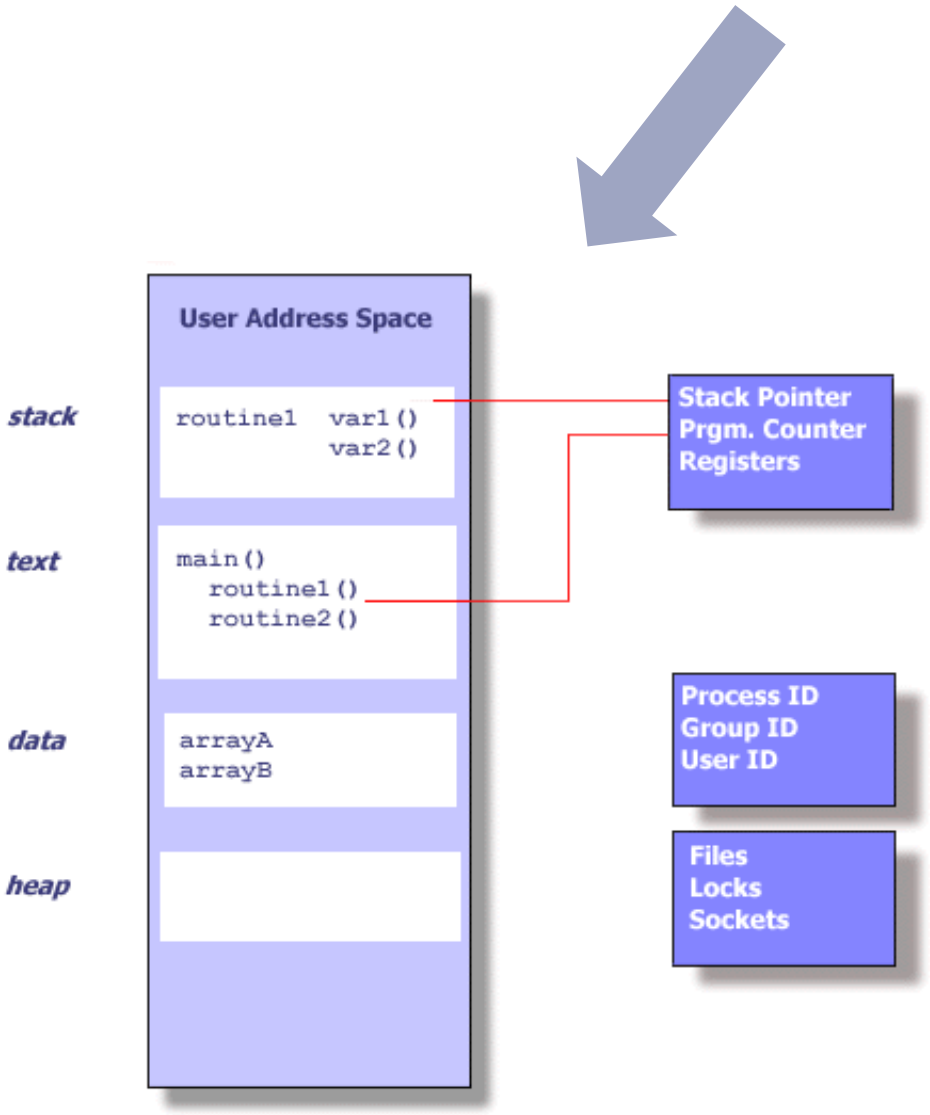
Threads First

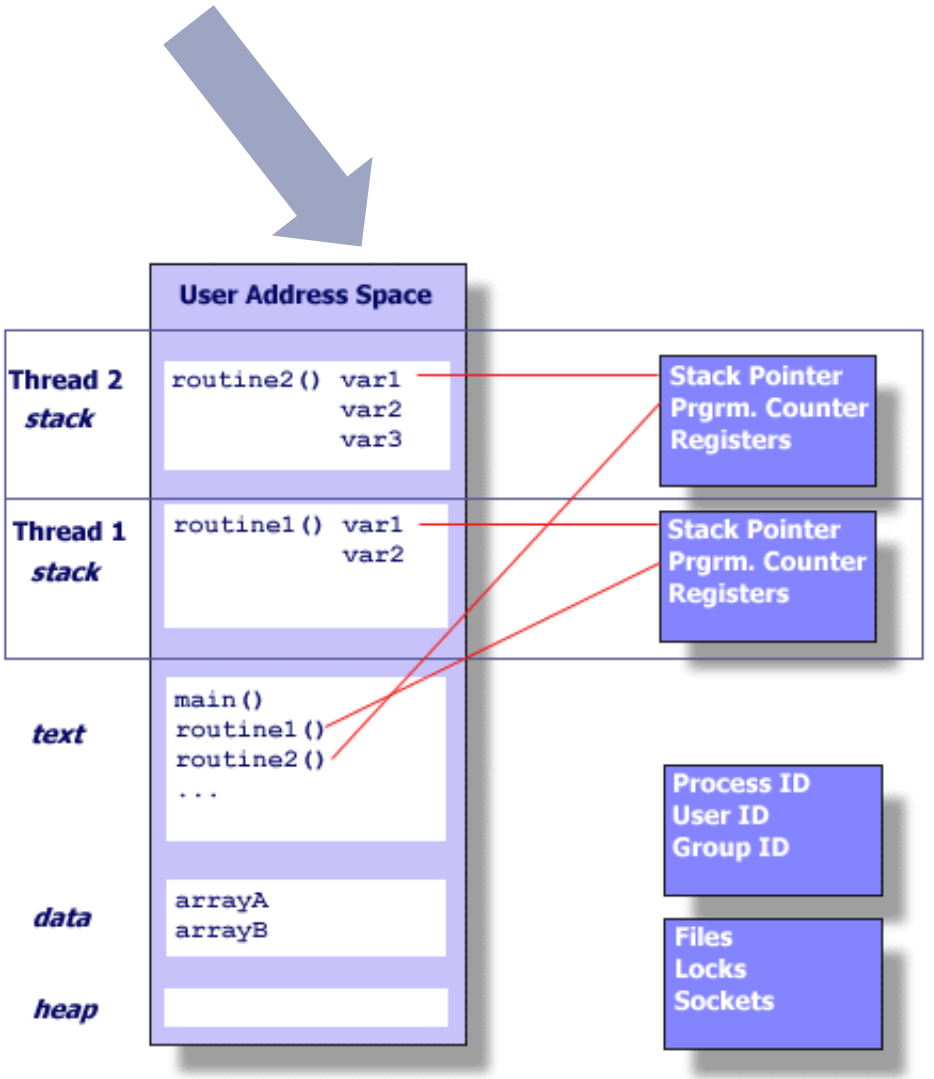
- Outline of Threads discussion:
  - What's a thread?
    - How many people have heard of / used threads before?
  - Coupling to architecture
  - Example: scheduling threads
    - Assume different architectural models
  - Programming models
  - Why intimate knowledge about HW is important

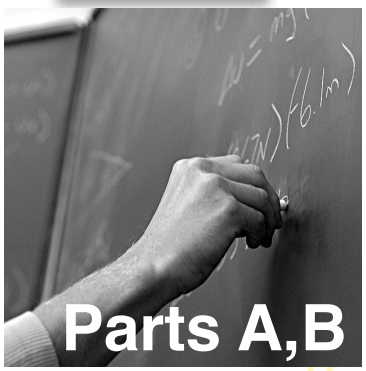
Processes vs. Threads

- Process
  - Created by OS
  - Much “overhead”
    - Process ID
    - Process group ID
    - User ID
    - Working directory
    - Program instructions
    - Registers
    - Stack space
    - Heap
    - File descriptors
    - Shared libraries
    - Shared memory
    - Semaphores, pipes, etc.
- Thread
  - Can exist within process
  - Shares process resources
  - Duplicate bare essentials to execute code on chip
    - Program counter
    - Stack pointer
    - Registers
    - Scheduling priority
    - Set of pending, blocked signals
    - Thread specific data

Processes vs. Threads

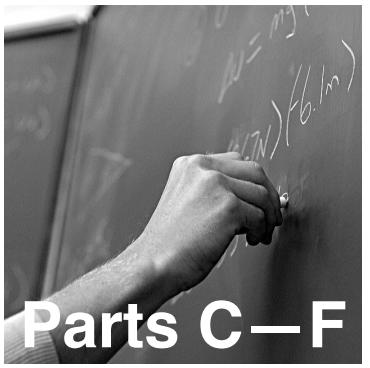




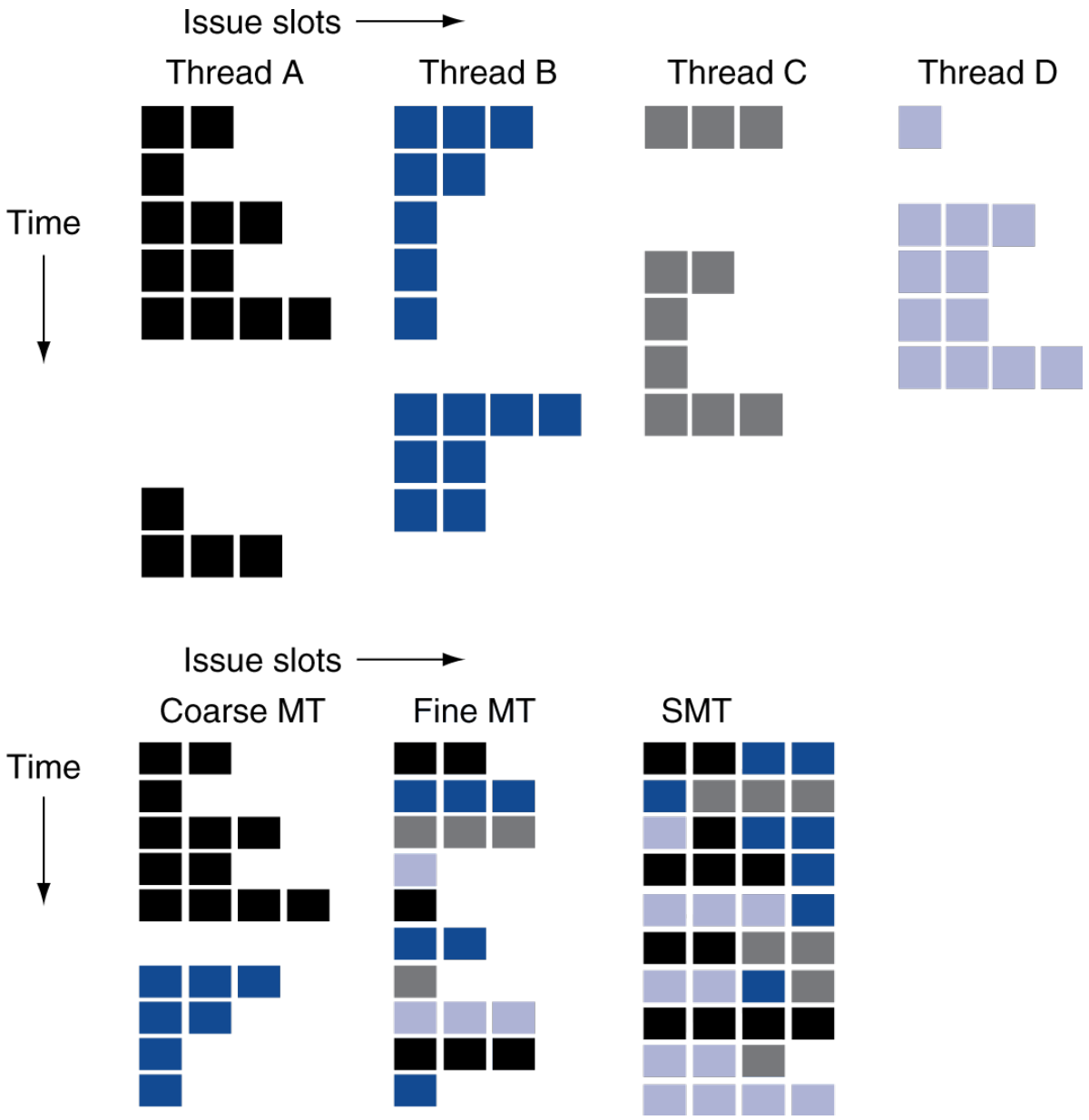


Multi-threading

- Idea:
  - Performing multiple threads of execution in parallel
    - Replicate registers, PC, etc.
  - Fast switching between threads
- Flavors:
  - Fine-grain multithreading
    - Switch threads after each cycle
    - Interleave instruction execution
    - If one thread stalls, others are executed
  - Coarse-grain multithreading
    - Only switch on long stall (e.g., L2-cache miss)
    - Simplifies hardware, but doesn't hide short stalls
      - (e.g., data hazards)
  - SMT (Simultaneous Multi-Threading)
    - Especially relevant for superscalar



Coarse MT vs. Fine MT vs. SMT



Mixed Models:

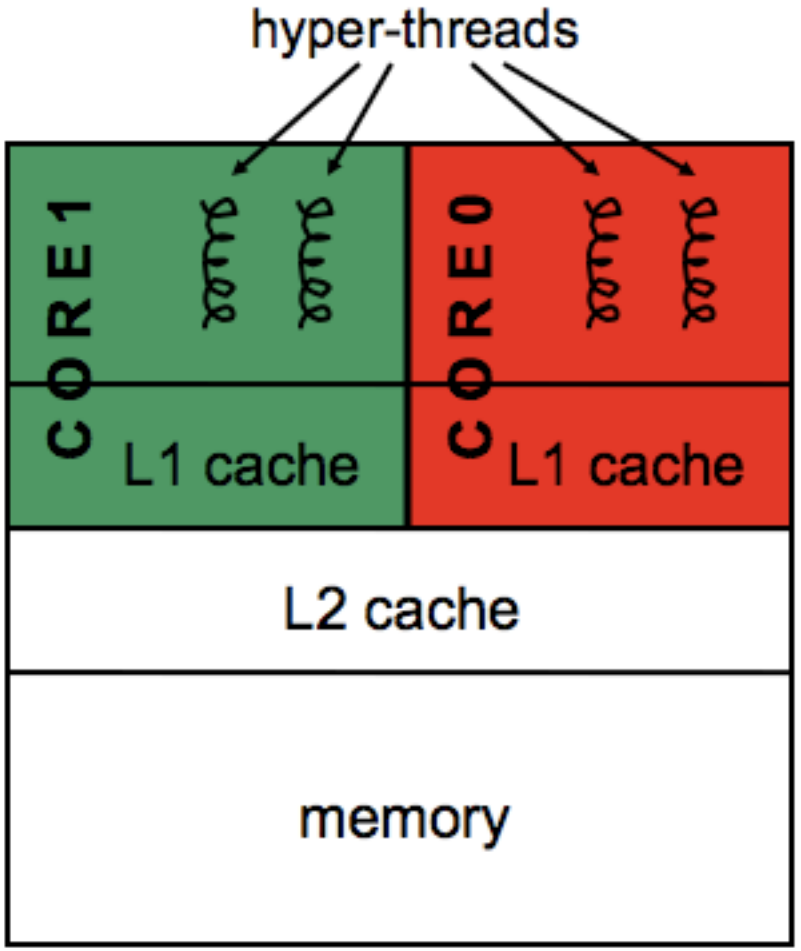
- Threaded systems and multi-threaded programs are not specific to multi-core chips.
  - In other words, could imagine a multi-threaded uni-processor too…
- However, could have an N-core chip where:
  - … N threads of a single process are run on N cores
  - … N processes run on N cores – and each core splits time between M threads

Comparison: multi-core vs SMT

- Multi-core:
  - Since there are several cores, each is smaller and not as powerful (but also easier to design and manufacture)
  - However, great with thread-level parallelism
- SMT
  - Can have one large and fast superscalar core
  - Great performance on a single thread
  - Mostly still only exploits instruction-level parallelism

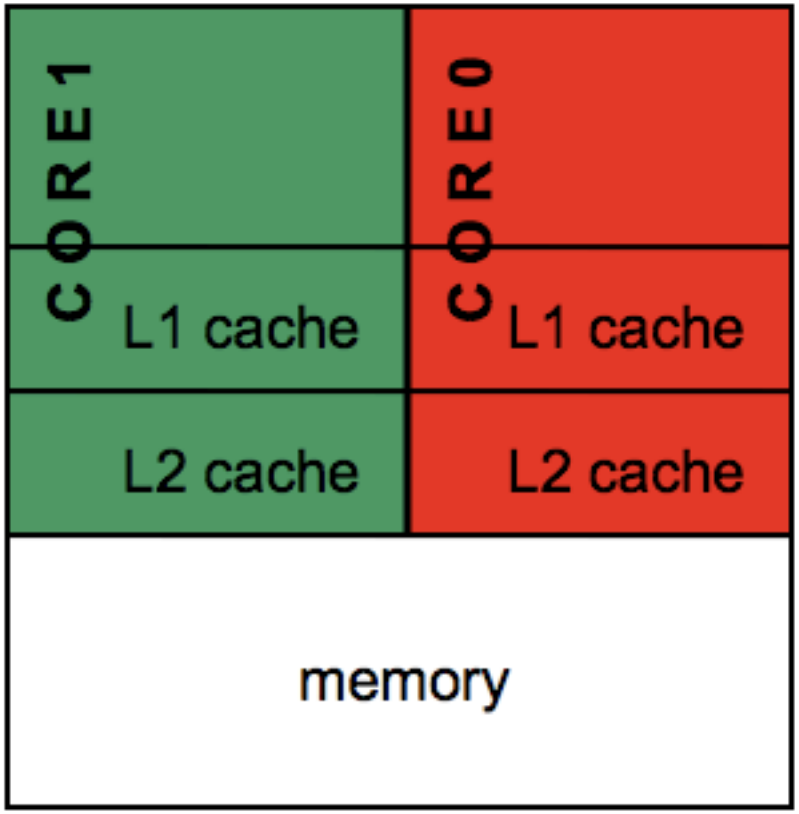
Or can do both…

- Dual-core
  Intel Xeon processors
- Each core is
  hyper-threaded
- Private L1 caches
- Shared L2 caches



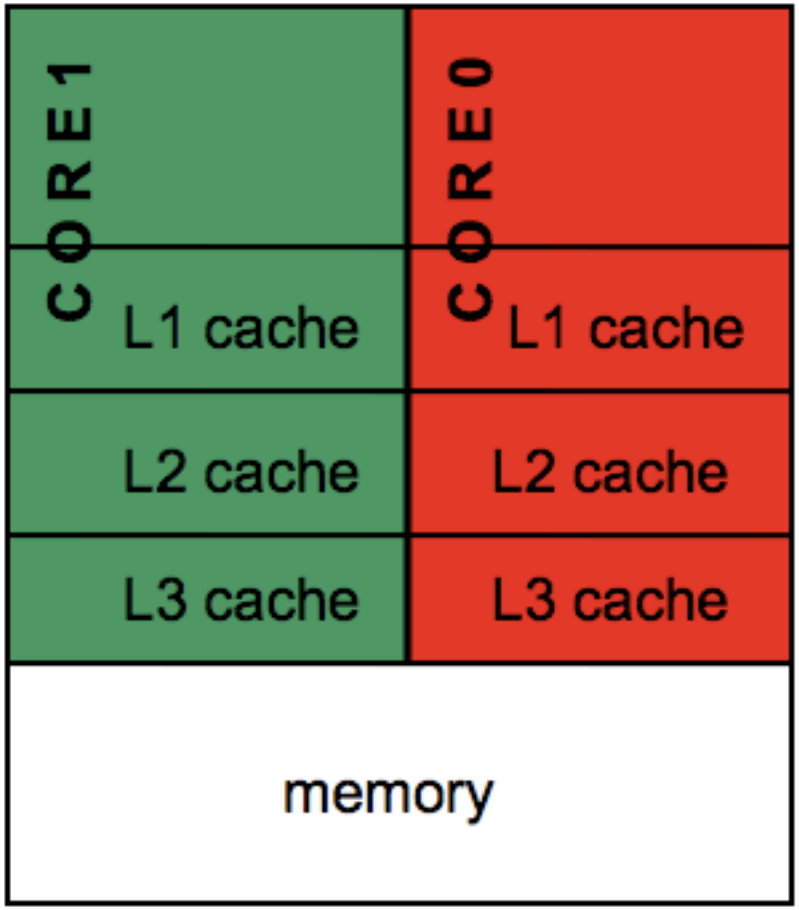
Real life examples…

Designs with private L2 caches



Both L1 and L2 are private

Examples: AMD Opteron,
AMD Athlon, Intel Pentium D



A design with L3 caches

Example: Intel Itanium 2

Writing threaded programs for supporting HW

- For UNIX systems, a standardized, C-language threads programming interface has been specified by the IEEE – POSIX threads (or Pthreads)
- For a program to take advantage of Pthreads…
  - Should be capable of being organized into discrete, independent tasks which can execute concurrently

Writing threaded programs for supporting HW

- For UNIX systems, a standardized, C-language threads programming interface has been specified by the IEEE – POSIX threads (or Pthreads)
- For a program to take advantage of Pthreads…
  - Should be capable of being organized into discrete, independent tasks which can execute concurrently

Writing threaded programs for supporting HW

- …but generally, programs that have the following characteristics are well-suited for Pthreads:
  - Work that can be executed OR data that can be operated on multiple tasks at the same time
  - Have sections that will block and experience long I/O waits
    - i.e. while 1 thread is waiting for I/O system call to complete, CPU intensive work can be performed by other threads
  - Use many CPU cycles in some places, but not others
  - Must respond to asynchronous events
    - i.e. a web server can transfer data from previous requests and manage arrival of new requests
  - Some work is more important than others (priority interrupts)

Impact of modern processing principles
(Lots of “state”)

- User:
  - state used for application execution
- Supervisor:
  - state used to manage user state
- Machine:
  - state that configures the system
- Transient:
  - state used during instruction execution
- Access-Enhancing:
  - state used to simplify translation of other state names
- Latency-Enhancing:
  - state used to reduce latency to other state values

Impact of modern processing principles
(Total state vs. time)