

# **ECE 571 – Advanced Microprocessor-Based Design Lecture 5**

Vince Weaver

`http://web.eece.maine.edu/~vweaver`

`vincent.weaver@maine.edu`

17 September 2019

# Announcements

- Homeworks
  - HW#1 graded
  - HW#2 due Thursday (a reading)
- Optional Readings
  - Pipeline Discussion: Computer Organization (RiscV) / Patterson and Hennesey  
Section 4.11 “Real Stuff: The ARM Cortex-A53 and Intel Core i7 Pipelines”
  - Power/Energy: Computer Architecture / Hennesey



and Patterson

Section 1.5 “Trends in Power and Energy in Integrated Circuits”



# HW#1 Review

- bzip2 benchmark – what does it do?
- 19 billion instructions +/- 1000 or so (this is test input maybe?)
- 13 billion cycles +/- 200 million (more variation than last year)
- Didn't ask, but roughly what's the IPC? 1.5 or so
- Reversed: similar – HW2 will show you why I asked that
- Perf record: 6.4s,

```
57.16%  bzip2      bzip2      [...] mainSort
17.57%  bzip2      bzip2      [...] BZ2_compressBlock
```



```

11.90%  bzip2      bzip2      [...] mainGtU.part.0
11.20%  bzip2      bzip2      [...] handle_compress.isra.2
 0.94%  bzip2      bzip2      [...] BZ2_blockSort

```

- Valgrind, 1m18s == roughly 20 times slower?

```

11,291,448,187  blocksort.c:mainSort  [/opt/ece571/401.bzip2]
 3,381,835,437  compress.c:BZ2_compressBlock  [/opt/ece571/401.bzip2]
 2,138,813,059  bzlib.c:handle_compress.isra.2  [/opt/ece571/401.bzip2]
 1,958,107,443  blocksort.c:mainGtU.part.0  [/opt/ece571/401.bzip2]
 165,396,105   blocksort.c:BZ2_blockSort  [/opt/ece571/401.bzip2]

```

- Gprof, also 4.3s  
 Different results, using function entry instead of exact instruction count for sampling?  
 Also, using older gcc, newer versions it's broken on x86\_64?

```

time      seconds      seconds      calls      s/call      s/call      name

```



70.77	2.59	2.59	53	0.05	0.05	mainSort
18.58	3.27	0.68	53	0.01	0.06	BZ2_compressB
8.20	3.57	0.30	12223	0.00	0.00	default_bzallo
1.09	3.61	0.04	53	0.00	0.05	BZ2_blockSort
0.82	3.64	0.03	1856468	0.00	0.00	add_pair_to_b

- Skid instructions – mov is more likely than sub?

```

|                                     n = ((Int32)block[ptr[unLo]+d]) - med;
1.17 | 5f0:  mov    (%r10),%edx
0.61 |      lea   (%rdx,%r13,1),%eax
1.11 |      movzbl (%r15,%rax,1),%eax
2.70 |      sub   %r9d,%eax
|                                     if (n == 0) {
1.08 |      cmp   $0x0,%eax

```

instructions:uppp

```

|                                     n = ((Int32)block[ptr[unLo]+d]) - med;
0.45 | 5f0:  mov    (%r10),%edx

```



```
0.86 |      lea    (%rdx,%r13,1),%eax
3.53 |      movzbl (%r15,%rax,1),%eax
1.25 |      sub    %r9d,%eax
      |          if (n == 0) {
1.15 |      cmp    $0x0,%eax
```



# Real-World Pipelining Examples (from P&H)

- ARM Cortex-A53 (found in Pi3)
  - Eight-stage pipeline
  - Dynamic multi-issue, two instructions
  - Static in-order pipeline
  - First 3 stages fetch two insns at a time, filling a 13-entry instruction queue (branch predictors)
  - Pipelines: one for load, one for store, two for ALU, one multiply, one divide, one FP/SIMD (mul/div/sqrt)





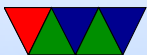
- one FP/SIMD for other
- What's the peak possible IPC?
  - Patterson and Hennessey report SPEC CPU 2006 INT results. Best case is hmmer (search for gene sequence) with IPC 1.03 (CPI 0.97). Worst is mcf (public transit vehicle scheduling) IPC 0.12 (CPI 8.56). Mostly memory constrained.
  - In-order so depends a lot on compiler to get good performance.
  - 100mW (1 core at 1GHz)
  - Intel Core i7 920 (Nehalem, 2008)



- Decodes CISC instructions to micro-ops
- Can issue up to 6 micro-ops per cycle
- 14 pipeline stages
- dynamic out-of-order with speculation
- register renaming, useful with speculation, as no need to store snapshot to undo speculation, just mark the speculated register results as invalid
- Instruction fetch, fetches 16 bytes. If wrong, 15 cycle penalty
- Predecode instruction buffer – transform 16 bytes (x86 insns 1-15 bytes) into x86 insns



- 18-instruction instruction queue.
- Micro-op decode – three decoders handle decode of instructions that map to 1 uop. One other handles microcode engine that produces longer sequences, up to 4uops a cycle.
- Can also do micro-op fusion (fuse two different insns into one uops, such as cmp/branch)
- Micro-ops go ins a 28-entry uop buffer  
Loop Stream Detector – if code is in tight loop (less than 28 insns) it can execute from this buffer and not need to fetch.



- Instruction issue. Reservation station. Up to six uops can be issued
- Finished instructions go back to reservation station and retirement unit, wait to update register state when determined it is no longer speculative.
- Once instruction hits the head of the reorder buffer, instruction commits and is removed from re-order buffer
- Even though 6 uops can issue, only 4 can be finished a turn? What's the peak IPC? (4)
- Again, SPEC CPU. Best is libquantum  $IPC=2.2$  (CPI



0.44). Worst, again, mcf  $IPC=0.37$  ( $CPI=2.67$ )

- Where do the wasted cycles go? Stalls? But also mis-speculation where work is done and then thrown out.
- 130 Watts (2.66GHz)



# Power and Energy



# Definitions and Units

People often say Power when they mean Energy

- Energy – Joules, kWh (3.6MJ), Therm (105.5MJ), 1 Ton TNT (4.2GJ), eV ( $1.6 \times 10^{-19}$  J), BTU (1055 J), horsepower-hour (2.68 MJ), calorie (4.184 J)
- Power – Energy/Time – Watts (1 J/s), Horsepower (746W), Ton of Refrigeration (12,000 Btu/h)
- Volt-Amps (for A/C) – same units as Watts, but not same thing
- Charge – mAh (batteries) – need V to convert to Energy



# Power and Energy in a Computer System

*Power Consumption Breakdown on a Modern Laptop, A. Mahersi and V. Vardhan, PACS'04.*

- Old, but hard to find thorough breakdowns like this
- Thinkpad Laptop, 1.3GHz Pentium M, 256M, 14" disp
- Oscilloscope, voltage probe and clamp-on current probe
- Measured  $V$  and Current.  $P=IIR$ .  $V=IR$   $P=IV$ ,  
subtractive for things w/o wires
- Total System Power 14-30W
- Old: no LED backlight, no SDD, etc.





Modern results are from CUGR/REU student research.

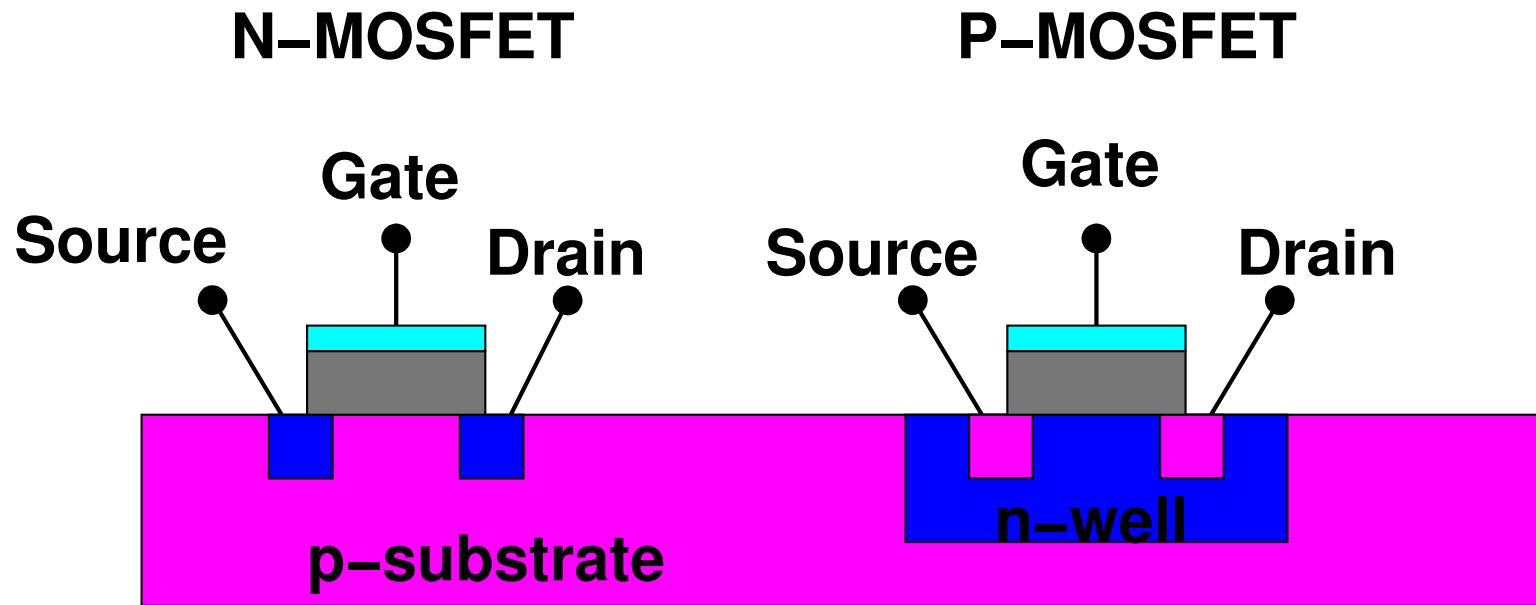
	Laptop (2004)	Modern	Server?
Hard Drive	0.5-2W	5W	
LCD	1W		
Backlight	1-4W		
CPU	2-15W	60+W	
GPU	1-5W	50+W	
Memory	0.5-1.5W	1-5W	
Power Supply	0.65W		
Wireless	0.1 - 3W		
CD-ROM	3-5W		
USB	(max 2.5W)		
USB keyboard		0.04W	
USB mouse		0.03W	
USB flash		0.5W	
USB wifi		0.5W	



# CPU Power and Energy



# CMOS Transistors



# CMOS Dynamic Power

- $P = C\Delta VV_{dd}\alpha f$

Charging and discharging capacitors big factor  
( $C\Delta VV_{dd}$ ) from  $V_{dd}$  to ground

$\alpha$  is activity factor, transitions per clock cycle

F is frequency

- $\alpha$  often approximated as  $\frac{1}{2}$ ,  $\Delta VV_{dd}$  as  $V_{dd}^2$  leading to  
 $P \approx \frac{1}{2}CV_{dd}^2f$

- Some pass-through loss (V momentarily shorted)



# CMOS Dynamic Power Reduction

How can you reduce Dynamic Power?

- Reduce  $C$  – scaling
- Reduce  $V_{dd}$  – eventually hit transistor limit
- Reduce  $\alpha$  (design level)
- Reduce  $f$  – makes processor slower



# CMOS Static Power

- Leakage Current – bigger issue as scaling smaller.  
Forecast at one point to be 20-50% of all chip power before mitigations were taken.
- Various kinds of leakage (Substrate, Gate, etc)
- Linear with Voltage:  $P_{static} = I_{leakage}V_{dd}$



# Leakage Mitigation

- SOI – Silicon on Insulator (AMD, IBM but not Intel)
- High-k dielectric – instead of  $\text{SiO}_2$  use some other material for gate oxide (Hafnium)
- Transistor sizing – make only the critical transistors fast; non-critical ones can be made slower and less leakage prone
- Body-biasing
- Sleep transistors

