The Future (past) of Computer Architecture

Sanjay Rajopadhye
Colorado State University

Outline

- CS270 recap: what's still a black box?
- Moore's law for ever
- The "walls" and surmounting them
  - Go for Speed
  - Bandwidth/Memory Wall
  - The decade of ILP and frequency
  - Power & Energy wall & the rise of multi-core
  - Utilization wall – next level or game over?
- Back to the present – PA6 details and issues

Colorado State University 2

What was brushed under the carpet in CS 270

- Combinational circuits are instantaneous
- Minimalist vs Efficient
  - LC-3 can execute any program
  - But does it do it efficiently?
- How fast can the machine go?
- How much power does it consume?
- What is the manufacturing cost?
  - Economies of scale

Colorado State University 3

Moore's Law

- Originally "formulated" by Gordon Moore (~1965) but other earlier observations too
  - The number of transistors that can be inexpensively placed on an integrated circuit is increasing exponentially, doubling approximately every years [later amended to 2 years]
  - http://en.wikipedia.org/wiki/Moore's_law
- It has held true till now and is expected to hold for about 10 more years

Colorado State University 4

Moore's Law corollaries

- For chip designers:
  - "we had better ensure that the exponential growth is maintained"
    - or else the competition will ☺
  - Main reason why the law is being sustained
- Other features of semiconductors & computing technology are also growing exponentially (but at different rates):
  - Frequency
  - Die size
  - Memory: density, and speed (bandwidth and latency)
  - Hard drive and I/O devices (both capacity and speed)
  - Networks
  - Power
  - Manufacturing cost

Colorado State University 5

Moore's law of expectations

- Better, faster, cheaper, lighter, …
- We as computer scientists are providing the technology that is changing the world exponentially
- Hard challenges
- Exciting potential
- Always room to innovate – if you stop learning you stagnate

Colorado State University 6

Exponential growth implies …

- When two quantities grow exponentially, but at different rates, their ratio also grows exponentially. Consider,

$$y_1 = a^x,$$

$$\text{and } y_2 = b^x \qquad \text{for } a \geq b \geq 1$$

$$y = \frac{y_1}{y_2} = \left(\frac{a}{b}\right)^x = \alpha^x \text{ for } \alpha \geq 1$$

Colorado State University 7

Moore's law walls

- Memory gap/wall:
  - Memory bandwidth grows much more slowly than processor speeds (since mid 80s, this was addressed by ever increasing on-chip caches).
- ILP (instruction level parallelism) wall
  - One way to exploit increased clock frequency was to increase the instruction-level parallelism on chip (deeply pipelined, out-of-order, VLIW, etc.) leading to complicated control logic.
- Power wall
  - Power dissipation ability is also increasing exponentially, but at such a slow rate that it has effectively peaked.
  - Power dissipation ability is also increasing exponentially, but at such a slow rate that it has effectively peaked.
- Utilization wall
  - Multi- and many-core trend cannot be sustained: there is no way to keep all the transistors on future chips active at all times, and this "dark silicon" will also be increasing (exponentially)

Colorado State University 8

Improving the speed

- Increasing the frequency
  - Simply riding More's law
- Better architecture through instruction level parallelism (ILP)
  - Pipelining
  - Super-scalar
  - Out-of-order execution

Tackling the memory wall

- Caches (since mid eighties)
  - If access to memory is slow, then build a faster (smaller) memory on-chip
- Need to exploit
  - Locality of reference
  - Reuse of data
  - Collaboration between architecture, compiler and operating system

Tackling the power wall

- Increasing the frequency implies increasing the heat generated
  - And that has to be dissipated (or else th chip will melt
- If you can't increase the frequency (raw speed)
  - Add more processors (cores)
  - "The processor is the new transistor" in Moore's law

Back to the present (PA6)

- Revisiting many of the concepts seen earlier:
  - HW4, LC3Viz and the "cycle-by-cycle" details of the instruction execution
  - Appendix A of the textbook
  - PA2 (32-bit floating point addition) in C
  - PA5 (16-bit FPADD) in assembly
- Code comprehension
  - Reading code written by others

Revisit normalization

- In scientific notation with 4-digit magnitude and 1-digit exponent:
  - What is the largest number:
    - 9.999E9
  - What is the smallest (positive) number?
    - 1.000E-9
- Can we do better?
  - Don't force numbers to be (always) normalized:
    - 0.001E-9

Colorado State University 13

For binary numbers

- Deal with the "implicit 1"
  - Sometimes it's there and sometimes it isn't
- When?
  - If exponent is 00000, no implicit 1
- Need to handle corner cases in the algorithm.
  - What if the inputs are a mix of normal and sub-normal numbers
  - What if the answer is?

Colorado State University 14