From Yale-45 to Yale-90: Let Us Not Bother the Programmers

Guri Sohi
University of Wisconsin-Madison

Celebrating Yale@75
September 19, 2014

Outline

- Where have we come from
- Where are we are likely going

Where From: Hardware

- Primary goal was performance
- Continuing increase in performance without demands on software
- Lots of “under the hood” innovations in cores (e.g., Big MF branch predictors)
  - Key enabling technique was sequential appearance and precise exceptions

Where From: Hardware

- Put more on core to achieve certain objective
  - Argument is “improve efficiency”
  - Multimedia, vectors, 64-bit, etc.
  - Incremental cost
  - Cores have become a “catch all”
  - Good for all, but not the most efficient for any
- Efficiency became important
  - Emergence of more efficient, special-purpose solutions (e.g., GPUs)

Where From: Software

- Few applications, few customers
- “Shrink Wrap” software: few applications and lots of customers
- Ubiquitous software: lots of diverse applications and lots of software

Where From: Software

- No worries when everything "under the hood"
- Significant challenges with multicore
  - Need to parallelize
- If biting the bullet, might as well go all the way
  - E.g., GPUs
- But mostly avoid difficulty and embrace convenience
  - Even if inefficient

Important Lessons

- When transistor budgets exceed certain amounts, the importance of certain techniques decreases, making room for other techniques
- Relative importance of special techniques diminishes over time
- Convenience key to software proliferation
- Mass volumes drive end result

Future Academic Research

- General-purpose App processing Units (GPAPUs)
- XY-DRAM
- 4D integration
  - Heterogeneity (XY-DRAM)
  - Dynamically varying distance between 3D layers
- Revisit everything (e.g., cache design and DRAM scheduling) with 4D integration with GPAPUs

Future

- Primary design goal: energy
- Hardware: use more transistors to save energy
- Software: keep doing things “under the hood”

Future

- Novel uniprocessor cores
- Lower energy devices
  - prone to errors
- Customized computation energy reducers (a.k.a. accelerators)
  - If can use software library, why use on multiple CPUs? Why not on customized hardware?

Processor Usage

- Have OS core, user core
- Have core that can only run 32/64-bit code
  - Specialization for 32-bit operands
- Have core that doesn't support precise interrupts
- Many other forms of limited functionality cores
  - Improve performance
  - Reduce energy

Processor Usage

- Steady demultiplexing of what was done on a general-purpose core
- Computation spreading
  - OS/user
- Separating specific code to accelerators
- Other forms of stripping out functionality in general purpose core
- “Mostly general-purpose” core

Hardware Going Forward

- Multiple mostly general-purpose processing cores
  - dynamically specialized
- Some "special-purpose" hardware
  - For more efficient processing
- Over-provisioning: pool of available (i.e., powered on) resources might change frequently
  - Now called "dark silicon"
- Will need to be transparent to software

What all is needed?

- Develop picks and shovels
- What are the mechanisms to ease software use of diverse hardware?
- Are we going to have higher level of exceptions/restart?
- Does the microarchitecture need low level restart?
  - Precise/non-precise core