Microprocessors -- 10 Years Back, 10 Years Ahead

Guri Sohi

University of Wisconsin — Madison
URL: http://www.cs.wisc.edu/~sohi

Outline

- The enabler: semiconductor technology
- Role of the processor architect
- Microarchitectures of the past 10 years
- Microarchitectures of the next 10 years

The Enabler: Semiconductor Advances

- Shrinkage in feature size
  - more transistors
  - faster transistors
- Increasing die size
  - more transistors

SIA Roadmap

Year	1997	1999	2002	2005	2008	2011	2014
Tech. (nm)	250	180	130	100	70	50	35
Memory(bits)	64M	256M	1G	4G	16G	64G	256G
Logic	3.7M	6.2M	18M	39M	84M	180M	390M

Source: Semiconductor Industry Association (SIA)

Role of Computer Architect

- Use available technology to perform processing tasks
  - exploit parallelism to extent possible
  - develop novel functionality to exploit parallelism
- Match processing tasks to hardware blocks constructed from available technology
- Do so in a manner that is easy to design/verify
- Get desired level of performance
  - time = number of instructions x cycles per instruction x clock cycle time

Architect's Role

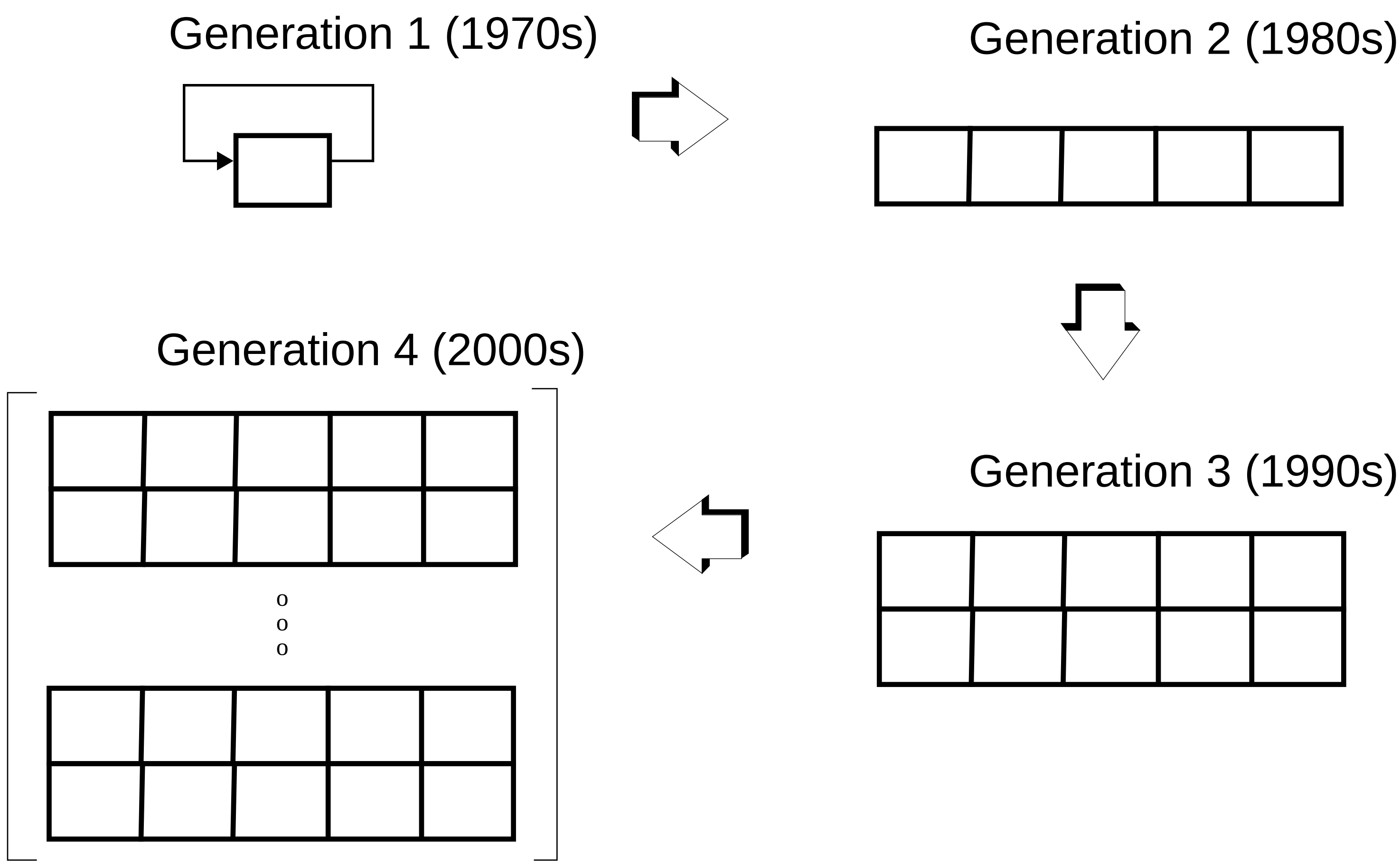
- Defining functionality
  - functionality to increase parallelism and its exploitation
  - functionality to deal with increasing latencies (e.g., caches)
- Implementing functionality
  - balancing various technology parameters
  - ease of design/verification/testing

The Performance Equation

Time = number of instructions x cycles per instruction x clock cycle time

- Not much can be done about first term in hardware
  - But, ...
- Logic speed increase decreases 3rd term
  - watch out for possible increase in 2nd term
- Use microarchitectural innovations to decrease 2nd and 3rd terms
  - exploit parallelism

Microprocessor Generations



Microprocessors -- 10 Years Back

- 30X increase in available transistors
  - how to use them?
- Little change in software programming model (still write programs in sequential languages
- Failed promise of automatic parallelization
- Great investment in existing software

Resort to low-level, instruction level parallelism (ILP)

Instruction Level Parallelism (ILP)

- Determine small number (10-40) instructions to be executed
  - control dependences (branches) hinder determination
- Determine dependence relationships and create dependence graph
  - dependence relationships determine shape of graph
- Use dependence graph to execute instructions in parallel
- Can be done statically (VLIW/EPIC) or dynamically (out-of-order (OOO) superscalar

Limitations to ILP

- Branch instructions inhibit determination of instructions to execute: control dependences
- Imperfect analysis of memory addresses inhibits reordering of memory operations: ambiguous memory dependences
- Program/algorithm data flow inhibits parallelism: true dependences
- Increasing latencies exacerbate impact of dependences

Use speculation to overcome impact of dependences

Speculation and Computer Architecture

Speculation: ".. to assume a business risk in hope of gain"

-- Webster

- Speculation in computer architecture is used to try to overcome constraining conditions
  - constraints due to "unknowns"

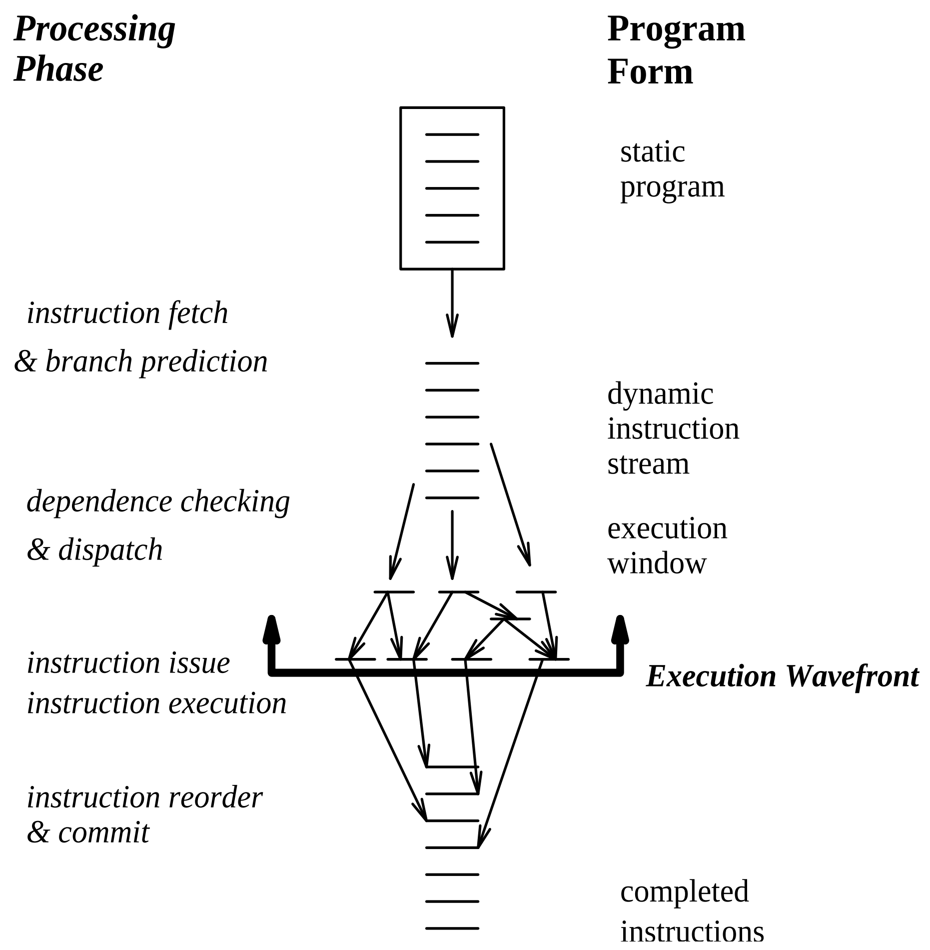
Speculation and Computer Architecture

- Speculate outcome of event rather than waiting for outcome to be known
  - mis-speculation if wrong
  - mis-speculation can have penalty
- Develop techniques to support speculation
- Develop techniques to speculate better

Control Speculation

- Predict outcome of branch instruction
- Speculatively fetch and execute instructions from predicted path
  - increase available parallelism
- Recover if prediction is incorrect

Model for Speculative Execution



Supporting Control Speculation

- Techniques to predict branch outcome: branch predictors
  - initiating speculation
  - improving accuracy of speculation
- Techniques to support speculative execution: reservation stations, register renaming, etc.
  - supporting speculative execution
- Techniques to give appearance of sequential execution: reorder buffers, etc.
  - doing it transparently

Basic mechanisms can support other forms of speculation as well

Performance-Inhibiting Constraints

- Control dependences: inhibit creation of instruction window
  - use control speculation
- Ambiguous data dependences: inhibit parallelism recognition
  - use data dependence speculation
- True data dependences: inhibit parallelism
  - use value speculation
- Common mechanisms may support different forms of speculation
- Different techniques to improve accuracy of speculation

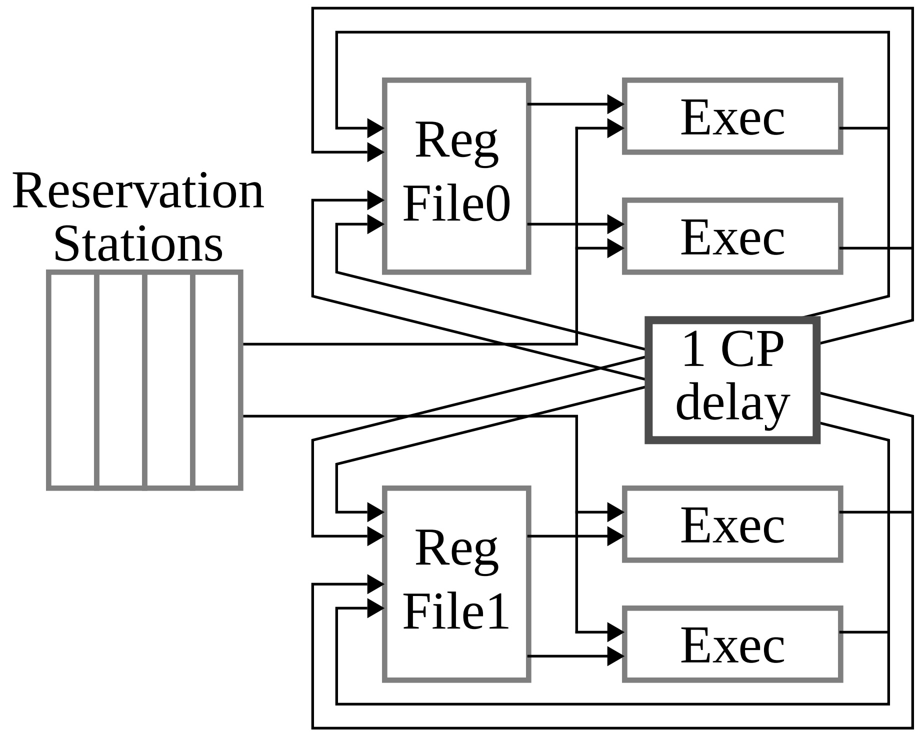
Microprocessors -- the Next 10 Years

- Factor of 30 increase in semiconductor resources
  - how to use it?
- New constraints
  - power consumption
  - wire delays
  - design/verification complexity
- New applications?
  - throughput-oriented workloads
  - (coarse-grain) multithreaded applications
  - dynamically-linked programs

Technology Trends

- Wires getting relatively slower
  - short wires for fast clock
  - implies increased latencies; localize communication to tolerate
- Design/verification for large numbers of transistors becoming unwieldy
- Power issues becoming very important

Alpha 21264


Reservation
Stations
Reg
File0
Exec
Exec
1 CP
delay
Reg
File1
Exec
Exec

Architect's Role Revisited

- Defining functionality
  - new models needed to further increase parallelism exploitation
- Implementing functionality
  - becoming a dominating factor?

Implications of Trends

- Implementation considerations will imply computing chips with multiple (replicated?) processing cores
  - a.k.a "multiprocessor" or "multiprocessor-like" or "multithreaded"
  - will start out as "logical" and will likely move towards "physical" replication
- How to assign work to multiple processing cores?
  - independent programs (or threads)
  - parts of a single program

Throughput-Oriented Processing

- Executing multiple, independent programs on underlying parallel microarchitecture
  - akin to traditional throughput-oriented multiprocessor
  - Significant engineering challenges, but little in ways of architectural/microarchitectural innovation

Can we use underlying "multiprocessor" to speed up execution of a single program?

Parallel Processing of Single Program

- Will the promise of explicit/automatic parallelism come true?
- Will new (parallel) programming languages take over the world?

Don't count on it!!!

Speculative Parallelization

- Sequential languages aren’t going away
- Use speculation to overcome inhibitors to “automatic” parallelization
  - ambiguous dependences
- Divide program into “speculatively parallel” portions, or “speculative threads”

Speculative Threads

- Subject of extensive research today
  - different thread types being discovered/investigated
  - control-driven threads
  - data-driven threads
- Several research examples (e.g., Wisconsin Multiscalar, Stanford Hydra)
- Two recent commercial examples
  - Sun Multithreaded Architecture for Java Computing (MAJC) -- circa 1999
  - NEC Merlot -- circa 2000

Generic circa 2010 microprocessor

- 4-8 general-purpose processing engines on chip
  - used to execute independent programs
  - explicitly parallel programs (when possible)
  - speculatively parallel threads
  - helper threads
- Special-purpose processing units (e.g., DSP functionality)
- Elaborate memory hierarchy
- Elaborate inter-chip communication facilities

Summary

- Semiconductor technology has, and will continue to, give computer architects new opportunities
- Architects have used speculation techniques to overcome performance barriers; will likely continue to do so
- Future microprocessors are going to have capability to execute multiple threads of code
- New models of speculation (e.g., thread-level speculation) will be needed to extract more parallelism