Beyond Claims of Free Transistors and Abundant Instruction-Level Parallelism

Michael D. Smith

STANFORD UNIVERSITY

Outline

- Current Superscalar Research Directions
- Available Instruction-Level Parallelism
- Transistor Cost and Hardware Complexity
- TORCH Architecture
- Summary

Current Research Trends - Part 1

- Find the Upper Bound on Instruction-Level Parallelism
  - Academic Papers

    "Limits of Instruction-Level Parallelism"
    - Wall (ASPLOS 1991)

    "Single Instruction Stream Parallelism Is Greater than Two"
    - Butler et al. (ISCA 1991)
  - Panel Sessions

    "Five Instructions Per Clock: Truth or Consequences"
    - HOT Chips III (1991)
  - rst Research Done in
    - Early 1970s - Riseman and Foster
    - 1984 - Nicolau and Fisher

Potential Performance Benefits

- Pre-1988 Reported Speedups of Superscalar Processors

Weiss and Smith (1984)	1.58
Sohi and Vajapeyam (1987)	1.81
Tjaden and Flynn (1970)	1.86
Tjaden and Flynn (1973)	1.96
Uht (1986)	2.00
Acosta et al. (1986)	2.79
Wedig (1982)	3.00
Kuck et al. (1972)	8
Riseman and Foster (1972)	51
Nicolau and Fisher (1984)	90

What Is the Potential Performance Benefit?

- Post-1988 Reported Speedups of Superscalar Processors

Horst, Harris, and Jardine (1990)	1.37
Wang and Wu (1988)	1.70
Smith, Johnson, and Horowitz (1989)	2.30
Murakami et al. (1989)	2.55
Chang et al. (1991)	2.90
Jouppi and Wall (1989)	3.20
Lee, Kwok, and Briggs (1991)	3.50
Wall (1991)	5
Melvin and Patt (1991)	8
Butler et al. (1991)	17+

The Answer

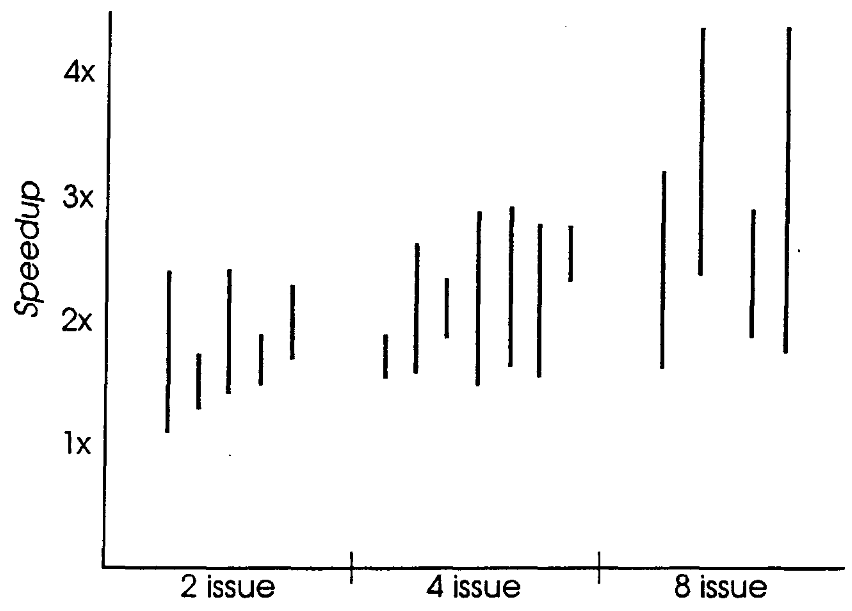
- The Upper Bound on Instruction-Level Parallelism is

BIG

- I'll even freely admit that it is greater than 2
- Also state that current studies have NOT reached the limit

  Reason: measuring programs scheduled for single issue

Correlating the Potential Studies



Current Research Trends - Part 2

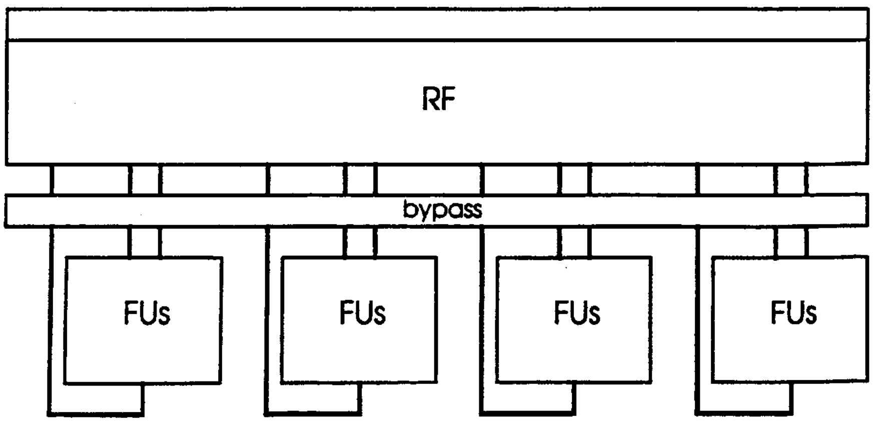
- Most Recent Studies Exploring Bigger and Bigger Machine Models
- Each Showing that "More Parallel Resources = More Performance"
  - Little to no consideration of implementation issues
- Trend Supported By Claims that "Hardware is free"
  - Unlimited transistor budget (in the near future)
  - Parallel designs minimally impact cycle time
- Claims Promote "Design for peak performance"
  - Free hardware negates implementation considerations

Transistors Are Free

- BUT hardware is NOT
- You can have as many individual transistors as you want
  - as long as you don't care about connecting them together
- Hardware cost is bounded by the cost of WIRES
- Cost(Wire) >> Cost(Transistor)
  - Wires are more costly in terms of
    - (1) area characteristics
    - (2) electrical characteristics
- Complexity cost dominated by interconnection cost

Wiring Dominates Big Issue Machines

- A general, 4-issue machine



- Bypass consists of 8 horizontal buses and very few transistors
- Complexity exists whether you see it directly or not

Complexity Considerations in Big-Issue Machines

- Operand bypassing grows as

$$\boxed{(\text{no\_of\_src\_operands})\,(\text{results\_in\_pipeline})\,i = 2\,(is)\,i \approx O(i^2)}$$

- Data dependence checks grow as

$$\boxed{(\text{no\_of\_src\_operands}) \sum_{n=1}^{i-1} n = 2\left(\frac{i(i-1)}{2}\right) \approx O(i^2)}$$

- Plus priority encoder complexity
  - additionally increases logic complexity or logic depth
- Complexity grows <u>faster</u> than issue rate

The Point

- Complexity is not free
- Complexity increases
  - the number of wires
  - the length of the wires
- Complexity increases with increasing issue rate
- Complexity affects performance
  - either run slower (due to naive design)
  - or increase your design time

The "Right" Approach

- Exploit Limited Parallelism with Rational Hardware
- Maintain the RISC Philosophy
    - Implementation complexity is important
    - "Relegate the Interesting Stuff to the Compiler" - Mike Powell
- Obtaining Cost-Effective Performance Is
    - More than pure dynamic scheduling
    - More than pure static scheduling
- Approach: Exploit Strengths While Minimizing Shortcomings

TORCH

- Good Utilization of Modest Hardware
    - Small issue rate - 2 instructions per cycle
    - Minimize expensive resources - 1 data memory port
    - Maximize expensive resource utilization - 2 integer ALUs
- Sophisticated Instruction Scheduling in Compiler
    - Allow compiler to schedule code speculatively - Boosting
    - Produce out-of-order execution with in-order issue hardware
    - Maintain look of in-order execution
- Plan for Software Compatibility
    - Superset of existing RISC architecture - MIPS compatible

Boosting

MIPS Code

```
     lb   $1,0($2)
     add  $2,$2,1
     ble  $1,90,L1
     add  $5,$1,-32

     sb   $5,0($3)
     j    L2
     add  $3,$3,1

L1:  sb   $4,0($3)
     add  $3,$3,1
     sb   $1,0($3)
     add  $3,$3,1
L2:
```

TORCH Code

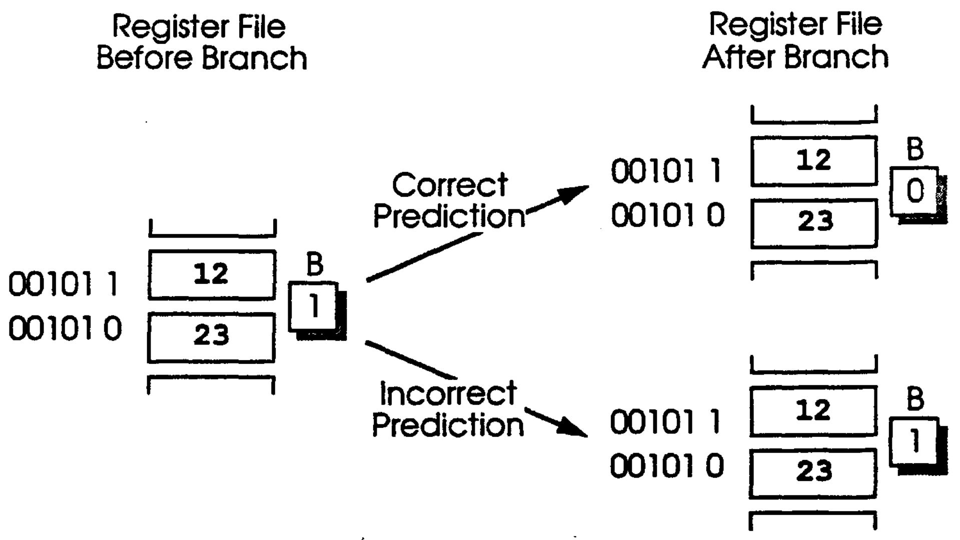
```
     lb      $1,0($2)          add  $2,$2,1
     sb.B    $4,0($3)          add  $3.B,$3,1
     ble.T   $1,90,L1          sb.B $1,0($3.B)
     add     $3.B,$3.B,1       add  $5,$1,-32

     sb      $5,0($3)          add  $3,$3,1
L1:
```

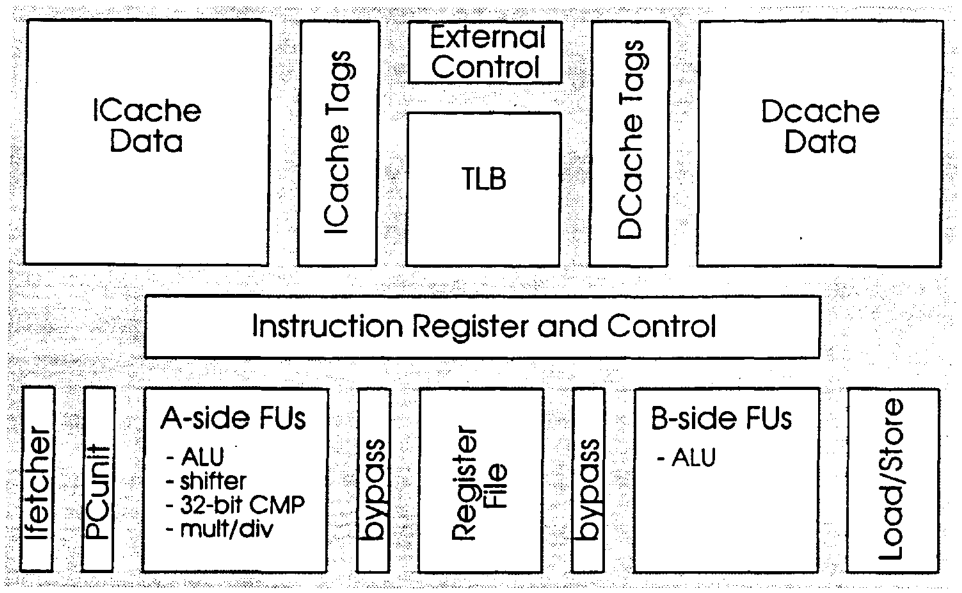
- Boosting removes control dependence
- Boosted results are
  - commited on correct prediction
  - squashed on incorrect prediction
- Boosting supports precise exceptions

Hardware Support for Boosting

- Shadow structure for register file, store buffer, exception logic



TORCH Floor Plan



Conclusions

- Build superscalar processors for "FT" and "AILP" - wrong frame of mind
  - only things free and abundant are opinions and points of view (well most people's opinions are free, sometimes too free)
- Need to look beyond "Limit Studies"
  - find cost-effective ways to use resources and parallelism
  - ultimate goal of Stanford's TORCH project
- Tackle hard problems
  - cheaply increase memory bandwidth
  - eliminate branches and branch effects
  - minimize global communication and buses

Acknowledgments

Mark Horowitz and Monica Lam

Tom Chanak, Phil Lacroute, John Maneatis,
Don Ramsey, Drew Wingard, and Victor Lam

STANFORD UNIVERSITY

Mike Johnson

ADVANCED MICRO DEVICES, INC.

Peter Davies and Earl Killian

MIPS COMPUTER SYSTEMS, INC.