

Designing Energy-Efficient Fetch Engines

Michele Co
Department of Computer Science
University of Virginia

Advisor:	Kevin Skadron
Co-Advisor:	Dee A.B. Weikle
Committee:	Jack Davidson (Chair) James Cohoon, John Lach, Christopher Milner



Overview

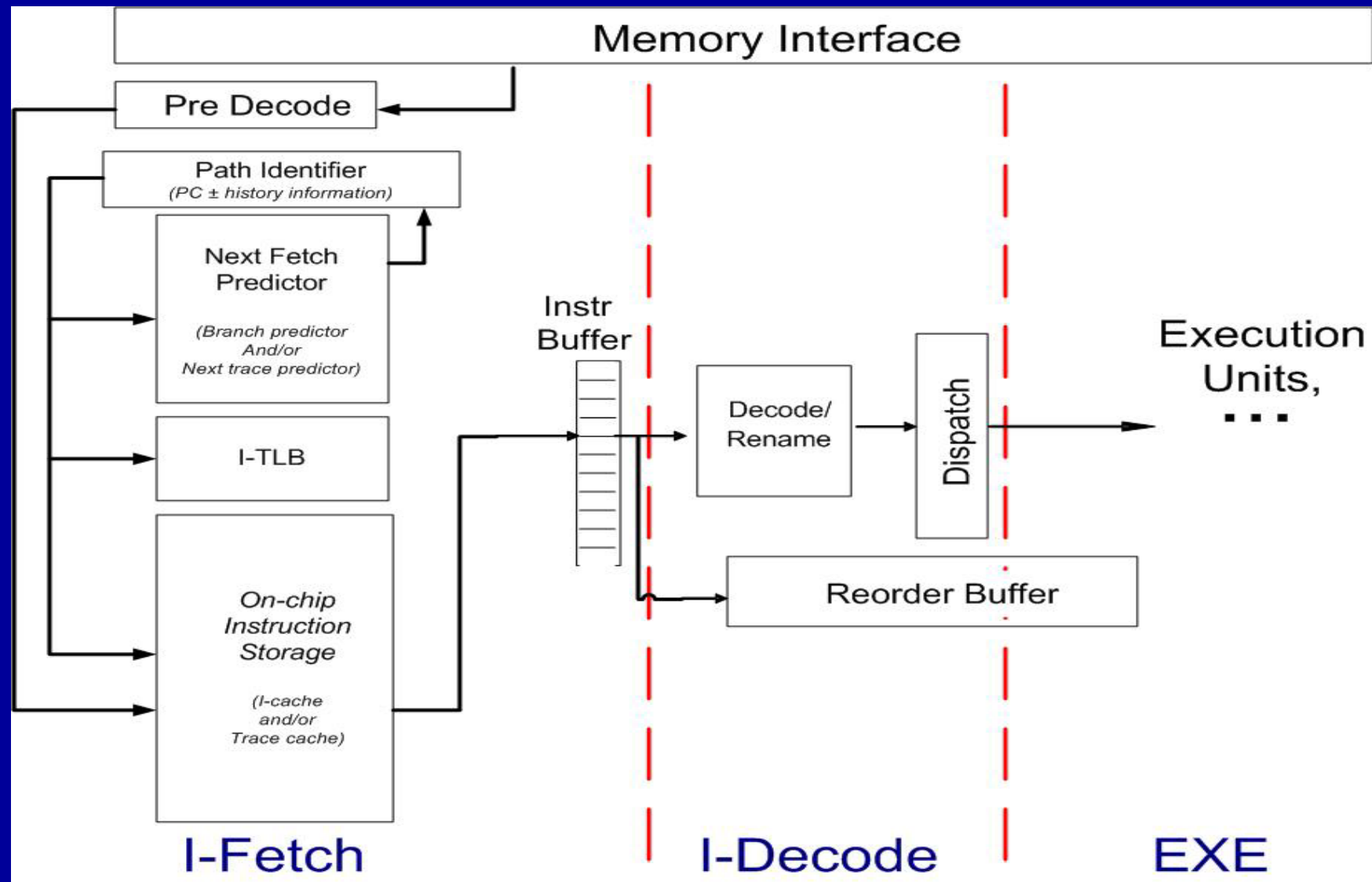
- Introduction
- Fetch Engine Design Space
- Related Work
- Results
- Summary



Introduction

- Energy efficiency
 - Balance power and performance (runtime)
- Fetch Engine
 - I-storage
 - I-TLB
 - Predictor



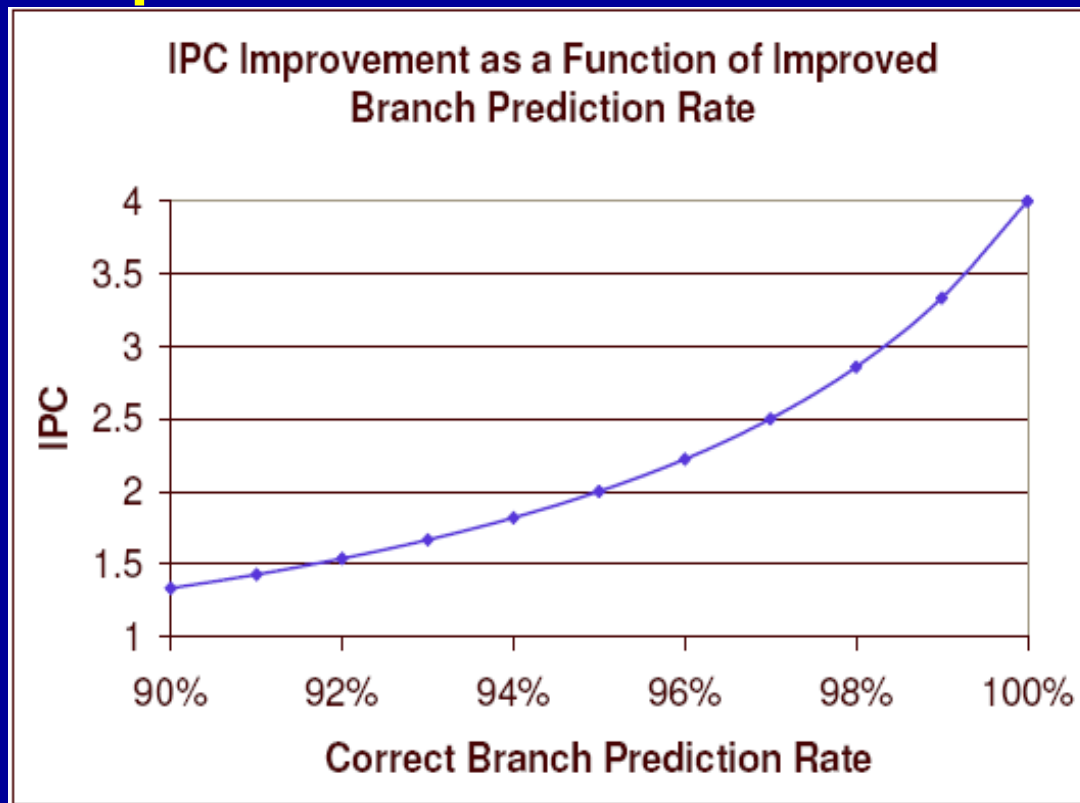


Fetch Engines are Important

- Provides instructions to execution units
 - Impacts overall processor energy
- Bottleneck to performance
 - Deep processor pipelines → high branch misprediction penalty
 - Mispredictions waste work and energy



Branch Prediction is Crucial to Improved Performance



- Branch prediction is the biggest limiter of performance [Jouppi & Ranganathan]



Why Study Fetch Engine Energy Efficiency?

- High fetch bandwidth mechanisms
 - Rotenberg, et al.; Black, et al.; many others
- Better branch predictor → better chip energy-efficiency
 - Parikh, et al.
- Recent branch predictors:
 - ↑ accuracy, ↑ performance, ↑ area
 - Jimenez; Seznec; Tarjan



What is the most energy efficient fetch organization?

- Performance design \neq Energy-efficiency design
- What improves energy efficiency?
 - Caches?
 - Branch predictors?
 - Other?

Thesis:

- Branch prediction is key factor affecting energy-efficiency
- Analytical methods are needed to help focus design space studies



Overview

- Introduction
- Fetch Engine Design Space
- Related Work
- Results
- Summary



Fetch Engine Design Space

- Design space parameters
 - Instruction storage order
 - Cache associativity and area
 - Instruction fetch bundle
 - Next fetch predictor
 - Instruction issue width
- Technology factors
 - Leakage power
 - Access latency
- System-level factors
 - Context switching



Research Goals

- Evaluate fetch engine design space for energy efficiency
- Develop techniques to aid in design space evaluations



Contributions

- Fetch engine design space evaluation for energy-efficiency
 - Insight: Branch prediction and access latency
- Ahead-pipelined next trace prediction
- Evaluated the effect of context switching



Contributions

- Breakeven branch predictor energy formulation
- Extension of breakeven formulation for in-order processors
- Evaluated potential for phase adaptation for branch predictors



Overview

- Introduction
- Fetch Engine Design Space
- **Related Work**
- Results
- Summary



Related Work

- Design space studies
 - Friendly, et al.; Rotenberg, et al. (trace caches)
- Power and energy consumption
 - Hu, et al.; (trace caches)
 - Bahar; Kim; Zhang (instruction caches)
 - Parikh, et al. (branch predictors)



Overview

- Introduction
- Fetch Engine Design Space
- Related Work
- **Results**
- Summary



Methodology

- Simulator and Power Models
 - SimpleScalar
 - Instruction cache and trace cache models
 - Branch predictors, next trace predictors
 - Context-switching
 - Wattch
- Benchmarks
 - SPECcpu1995, SPECcpu2000
- Metrics
 - Performance: Instructions per Clock (IPC), branch misprediction rate
 - Energy-efficiency: Energy, Energy-Delay-Squared Product (ED²)

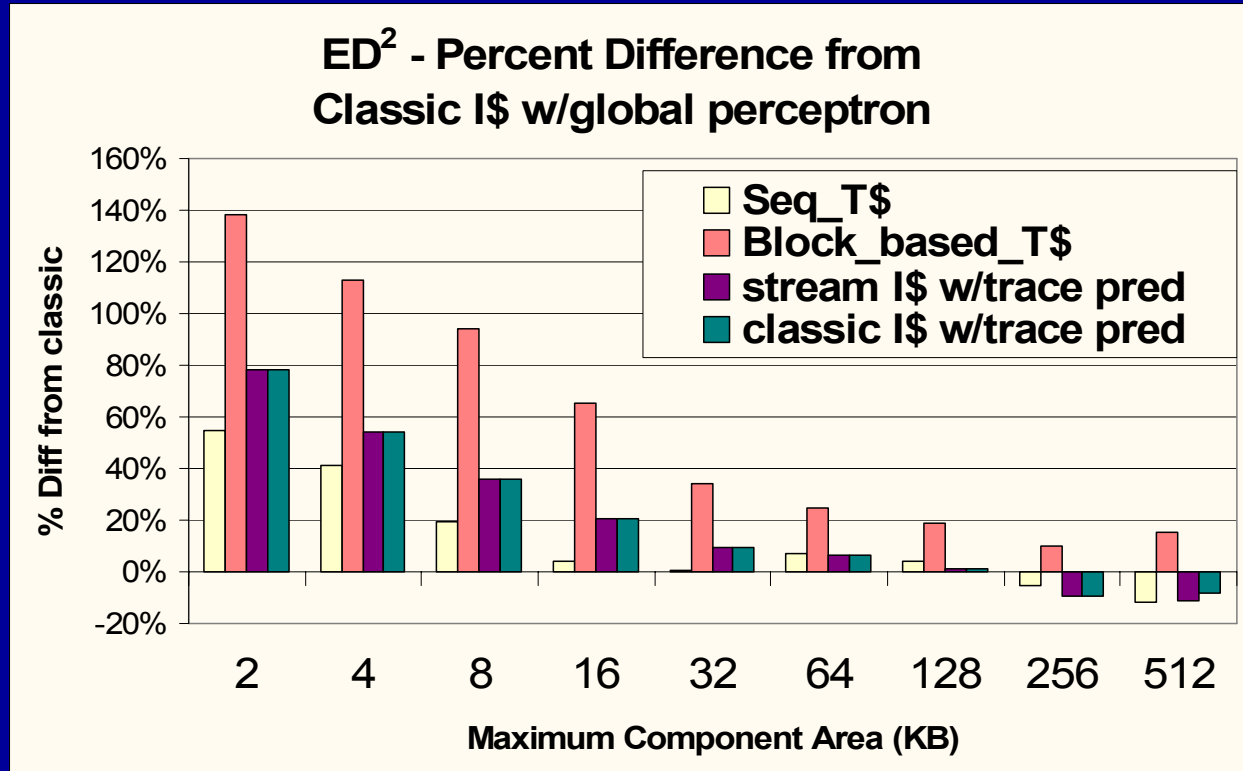


Fetch Engine Design Space Study

- Varied fetch organization
 - Sequential trace cache
 - Block-based trace cache
 - Instruction cache
 - Classic fetch
 - Streaming fetch
 - Branch predictor / trace predictor
- Varied component sizes
 - 2 KB - 512 KB



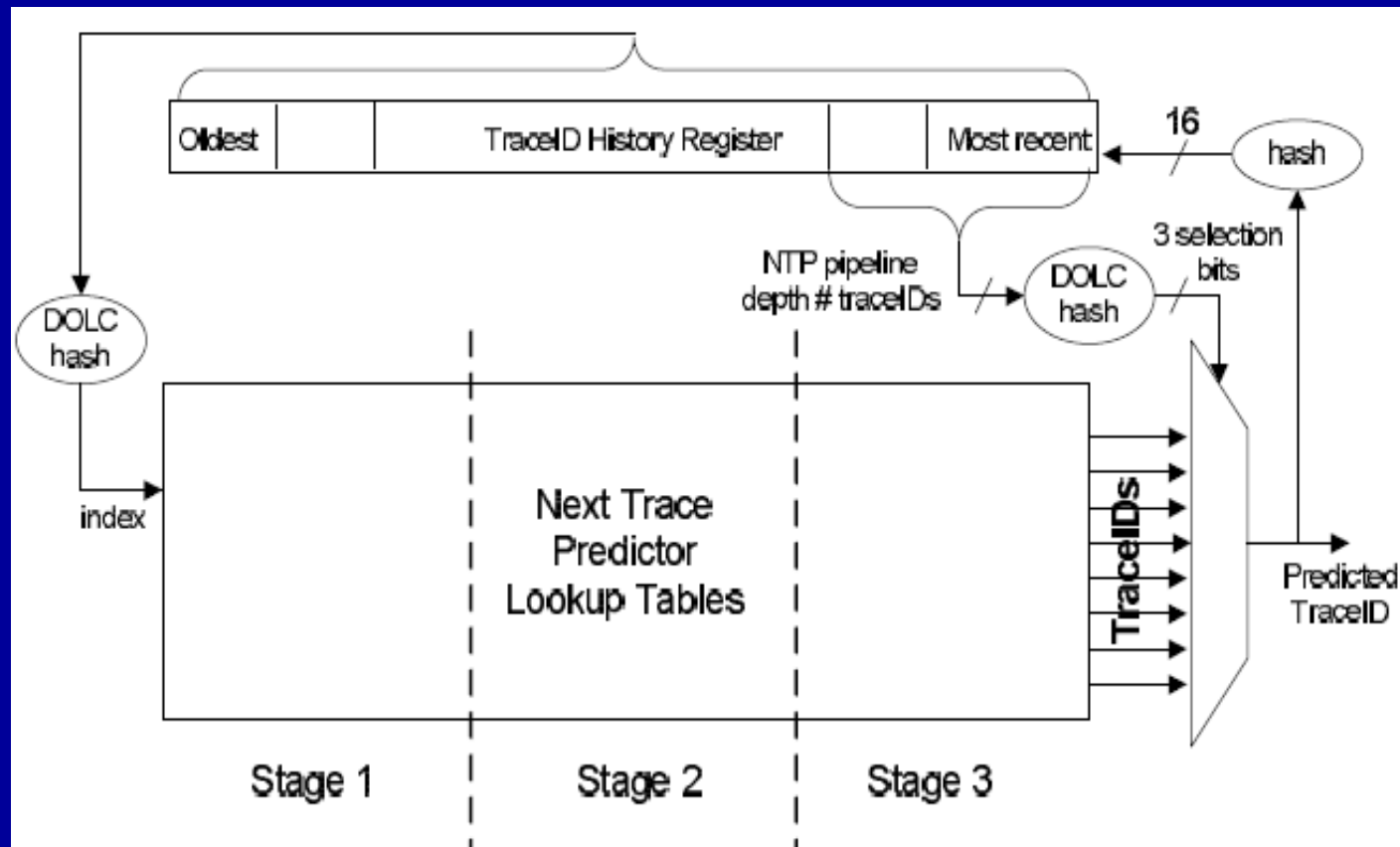
Fetch Engine Energy-Efficiency



- T\$ only energy efficient at large areas
 - Trace prediction constrained by small areas
- Branch prediction is determining factor!



Ahead-pipelining Next Trace Prediction



- 17.2% IPC improvement, 29% lower ED²



Branch Predictor Energy Budgets

- $E_{\text{total}} = E_{\text{bpred}} + E_{\text{remainder}}$
- $ED_{\text{new}}^2 \leq ED_{\text{ref}}^2$

Upper bound

$$E_{\text{bpred_budget}} \leq \frac{ED_{\text{ref}}^2}{D_{\text{new}}^2} - E_{\text{remainder_new}}$$

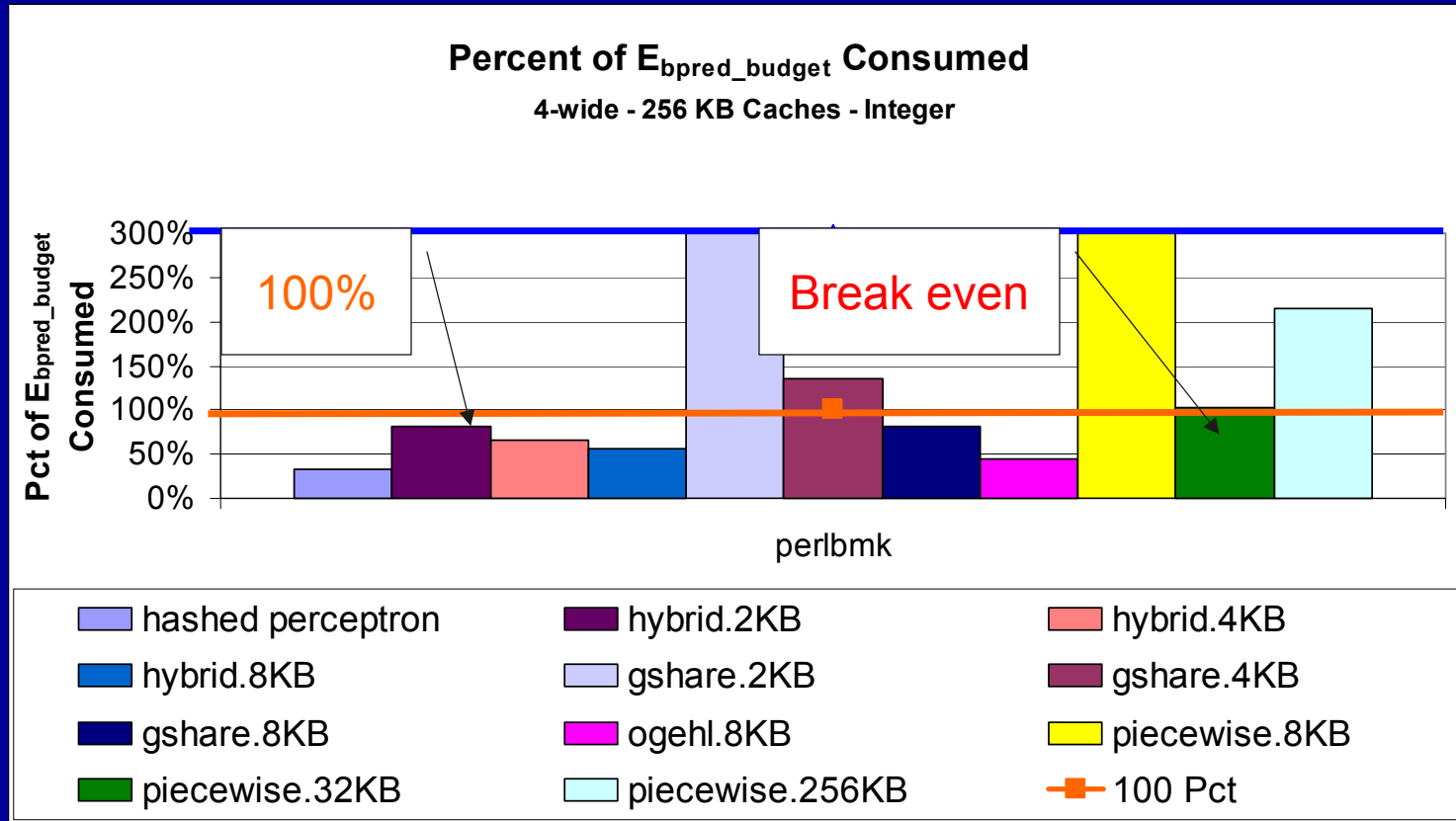


Branch Predictor Energy Budgets (Intuitively)

- How much energy may the branch predictor afford to consume to break even?



Comparing Bpred Energy Budgets



- $E_{\text{actual}} < E_{\text{budget}} \rightarrow$ More energy efficient

Branch predictors are not equally energy efficient for all programs



Breakeven Methodology

How well must a branch predictor perform in order to break even in energy-efficiency?

1. Choose an energy-efficiency metric

- Energy, ED^2

2. Develop a breakeven formulation using the chosen metric

- $E_{\text{new}} \leq E_{\text{ref}}; ED^2_{\text{new}} \leq ED^2_{\text{ref}}$



Step 3: Expand the Formulation

3. Represent formulation in terms of factors that don't require cycle-accurate simulation

- Base processor – power, misprediction penalty, clock frequency
- Program – number of instructions, branch density
- Branch predictor – power, misprediction rate

$$\left(P_{\text{rem}} + P_{\text{nc}} \frac{V_{\text{TP}} \cdot (MFD)}{f_{\text{BP}}} + P_{\text{BP}} \frac{MFD}{f_{\text{BP}}} + P_{\text{BP}} \frac{v_{\text{BP}}}{f_{\text{BP}}} * BD \right) T_{\text{new}} \leq P_{\text{ref}} T_{\text{ref}} \left(P_{\text{rem}} + F P_{\text{new}} \frac{v_{\text{BP}}}{f_{\text{BP}}} * BD \right)$$



Step 4: Solve the Formulation

4. Solve formulation for parameter of interest
 - Change in misprediction rate (MP_{Δ})

$$MP_{\Delta} \geq \frac{P_{bp} \left(bb_size + Rate_{MP_{without_bp}} MP_{penalty} \right)}{MP_{penalty} \left(P_{rem} + P_{bp} \right)}$$



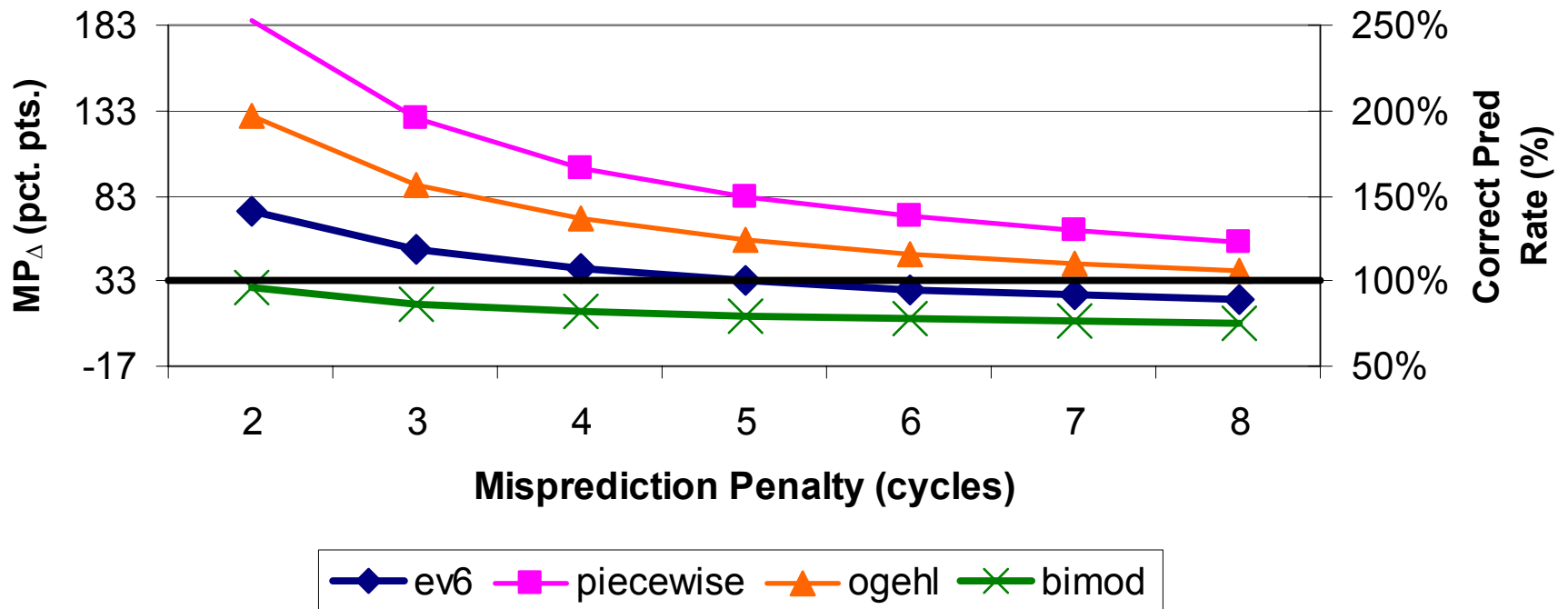
Step 5: Narrow the Design Space

5. Cull design space, eliminate non-interesting design points from cycle-accurate simulation evaluation



MP_Δ by Misprediction Penalty

Reference: Static BTFNT - Energy Breakeven
(330 MHz, Base Power: 1.85528109e-9 J/cycle)

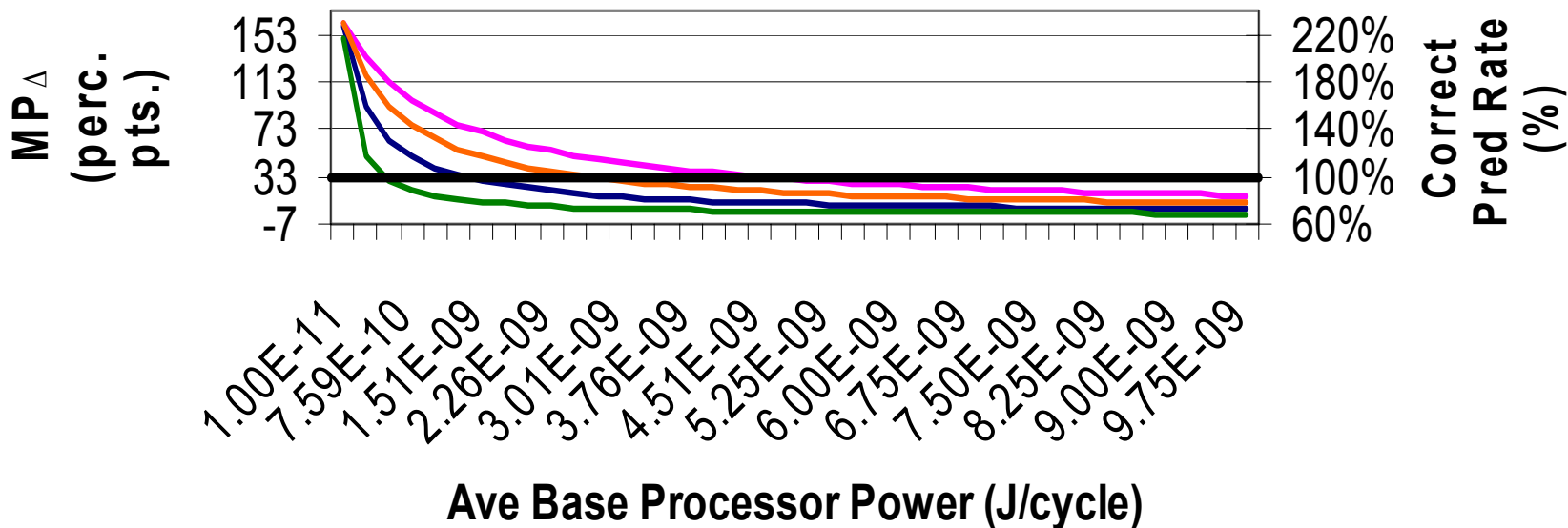


MP_Δ by Base Processor Power

Reference: Static BTFNT - Energy Breakeven

Misprediction Penalty: 7 cycles

(330 MHz, Base Power: 1.85528109e-9 J/cycle)



— ev6

— piecewise

— ogehl

— bimod

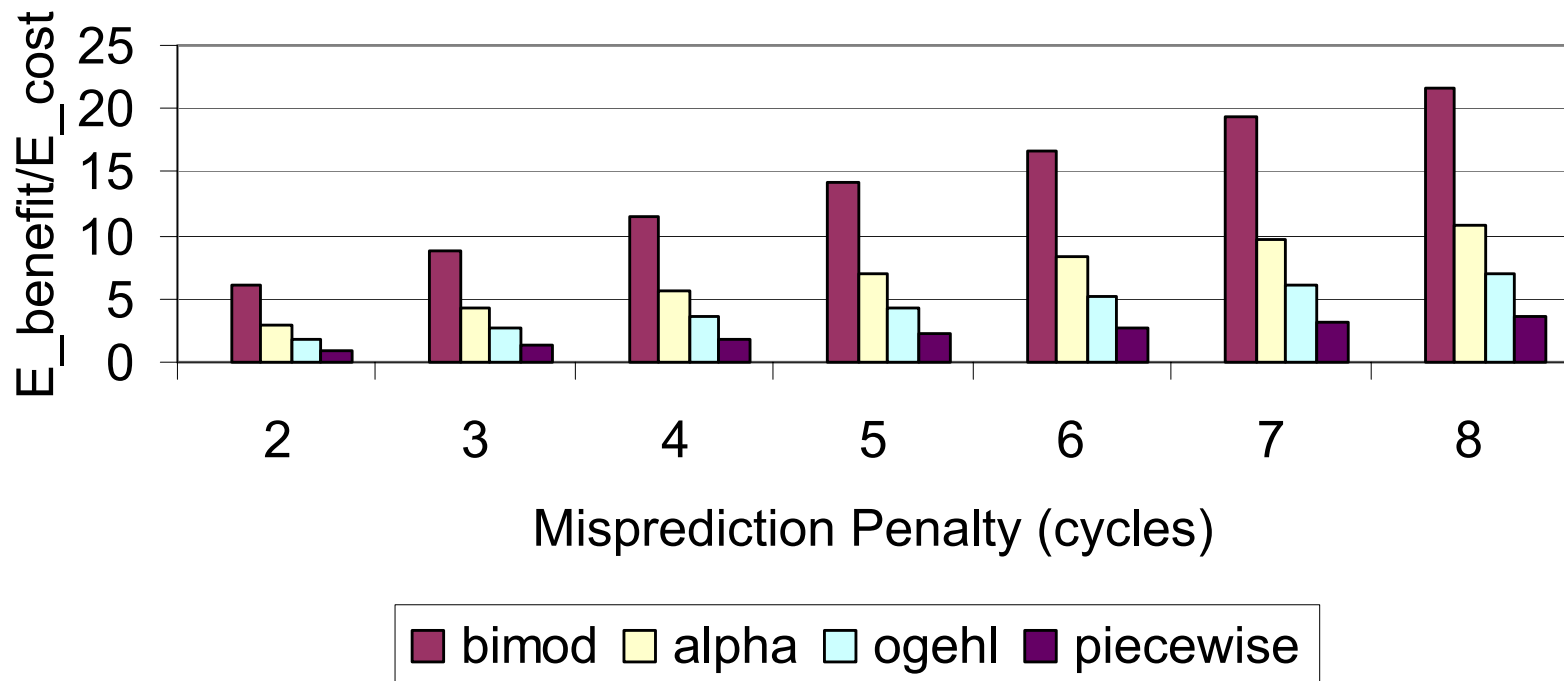


Branch Predictor Benefit/Cost

Energy Benefit/Cost Ratio

Reference: Static BTFNT

(330 MHz, Base Power: 1.85528109e-9 J/cycle)



Adapting the Fetch Engine for Energy Efficiency

- Proposed Idea
 - Adapt branch predictor based on program phase behavior
- Finding
 - Little potential for phase-based branch predictor adaptation
 - Given state-of-the-art branch predictors and the SPEC2000 benchmark suite



Overview

- Introduction
- Fetch Engine Design Space
- Related Work
- Results
- **Summary**



Contributions

- Fetch engine design space evaluation considering energy-efficiency [ACM TACO, *in review*]
- Ahead-pipelined next trace predictor [ACM TACO, *in review*]
- Effects of context-switching on branch predictors [ISPASS 2001]
- Branch prediction breakeven formulation [WCED 2005]
- Branch predictor benefit/cost formulation
- Potential for phase-based branch predictor adaptation



Future Work

- Fetch engine design space
 - More sophisticated trace selection heuristics
- Branch predictor breakeven formulation
 - Extend to consider more components
- Benefit-cost methodology
 - Extend to more complex processor designs
 - Incorporate into a semi-automated tool



Observations

- Trace caches and instruction caches have similar performance and energy-efficiency
- Branch prediction is critical to energy-efficiency
- Access latency is a critical limiter to branch prediction accuracy
- Techniques that attack this problem can help branch prediction accuracy (ahead-pipelined NTP)
- Realistic time slices have little effect on branch predictor accuracy



Observations (cont'd)

- Increasing leakage ratio does not affect relative energy-efficiency of fetch designs
- Design space studies are complex and time consuming. Analytical methods are needed to narrow the evaluation space.
 - Branch predictor breakeven energy budget
 - Benefit/cost formulation and metric



Designing Energy-Efficient Fetch Engines

Michele Co
Department of Computer Science
University of Virginia

Advisor: Kevin Skadron
Committee: Jack Davidson (Chair)
James Cohoon, John Lach,
Christopher Milner, Dee A.B. Weikle

