

# International Conference on Supercomputing (ICS 2002)

*New York City, USA, June 2002*

## Dual Path Instruction Processing

**Juan L. Aragón<sup>1</sup>, José González<sup>1,\*</sup>, Antonio González<sup>2,\*</sup> and James E. Smith<sup>3</sup>**

<sup>1</sup> Dept. Ing. y Tecnología de Computadores  
Universidad de Murcia

<sup>2</sup> Dept. d'Arquitectura de Computadors  
Universitat Politècnica de Catalunya

<sup>3</sup> Dept. Electrical and Computing Eng.  
University of Wisconsin-Madison

*\* Currently at Intel Barcelona Research Center*

**e-mail: [jlaragon@dittec.um.es](mailto:jlaragon@dittec.um.es)**

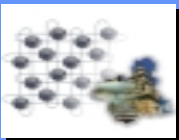


**GACOP**



# Motivation

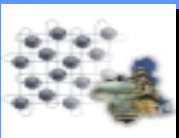
- ▶ **Two ways of reducing performance degradation due to branch mispredictions**
  - Improving prediction accuracy
  - **Reducing branch misprediction penalty**
- ▶ **Branch misprediction penalty**
  - Deeper pipelines cause higher misprediction penalties
    - Pentium 4 (20 stages); Power 4 (14 stages)
    - **Example: IPC slowdown of 22%**, using 32 KB *gshare* comparing a pipeline of 20 stages over 10 stages (*go*)



# Motivation

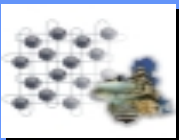
## ▶ Causes of performance degradation after a branch misprediction

- Pipeline must be squashed
- Many cycles until new instructions can be issued
  - Front-end length
- Instruction window is not full during many cycles
  - ILP cannot be fully exploited
- Correct instructions cannot be scheduled ahead a mispredicted branch



# Outline

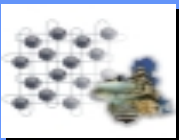
- ▶ **Misprediction Penalty Analysis**
- ▶ **Proposal**
- ▶ **Dual Path Instruction Processing (*DPIP*)**
- ▶ **Experimental Results**
- ▶ **Sensitivity Analysis**
- ▶ **Conclusions**



# Misprediction Penalty Analysis

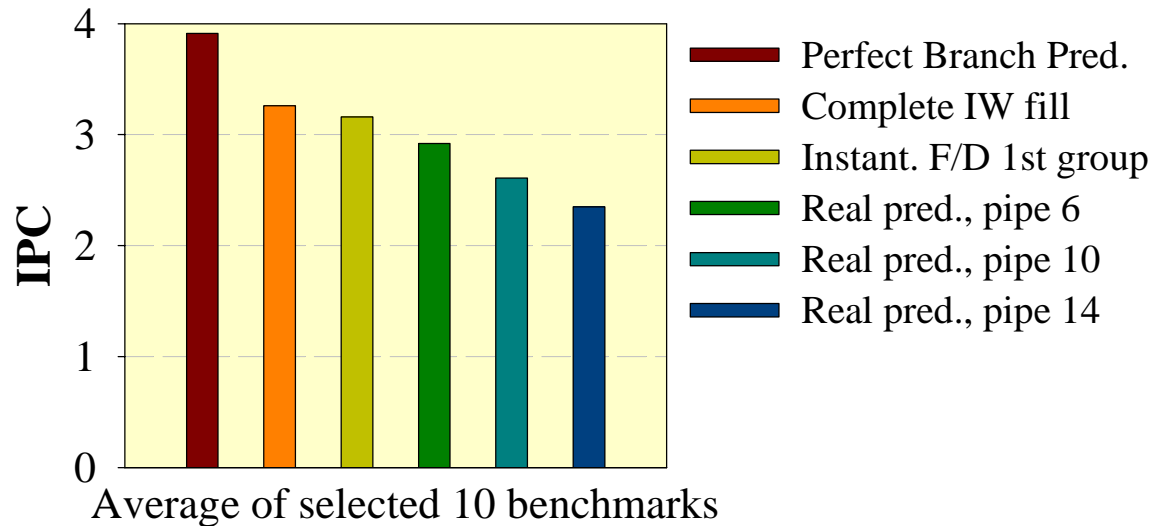
## ▶ Three Components

- Pipeline-fill penalty
  - Delay between the misprediction and the first correct instruction enters the window
  - Depends on
    - Pipeline length, Recovery actions
- Window-fill penalty
  - Window empty many cycles after misprediction
  - ILP cannot be fully exploited
- Serialization penalty
  - Correct instructions cannot be scheduled ahead of the mispredicted branch



# Misprediction Penalty Analysis

## ► Analysis of each component

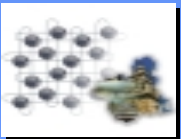


	Overall loss	Pipeline-fill penalty	Window-fill penalty	Serialization penalty
pipeline 6	25%	25%	10%	65%
pipeline 10	33%	44%	7%	49%
pipeline 14	39%	54%	6%	40%



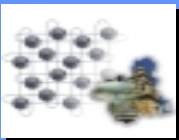
# Outline

- ▶ **Misprediction Penalty Analysis**
- ▶ **Proposal**
- ▶ **Dual Path Instruction Processing (*DPIP*)**
- ▶ **Experimental Results**
- ▶ **Sensitivity Analysis**
- ▶ **Conclusions**



# Proposal

- ▶ Reduce *Pipeline-fill* and *Window-fill* penalties
- ▶ Dual Path Instruction Processing (*DPIP*)
  - Fetches, decodes and renames both paths
    - Reduce *Pipeline-fill* penalty
    - Hide front-end stages
  - Alternative path instructions are pre-scheduled in an estimated execution order
    - Reduce *Window-fill* penalty
    - Similar effect as filling the window completely
- ▶ Confidence estimation
  - Used to filter branches that must be forked





# Related work

## ▶ Multiple path execution (*MPE*)

- Fetch, decode and *execute* instructions from multiple paths
  - *Selective Dual Path Execution* (Heil & Smith, Tech.Report'97)
  - *PolyPath* (Klauser *et al*, ISCA'98)
  - *Threaded Multiple Path Execution* (Wallace *et al*, ISCA'98)
- Too expensive (drawbacks)
  - Aggressive fetch engines (allowing up to 8 different paths!!!)
  - Bigger register files, instruction windows and ROB's
  - Complexity of selective flush
  - Resource contention: more functional units, memory ports,...
  - Energy consumption: resources used by useless instructions

*DPIP does **not** execute instructions*

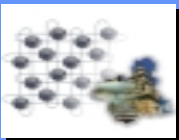


*balance between complexity, cost, and performance*



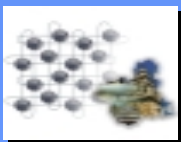
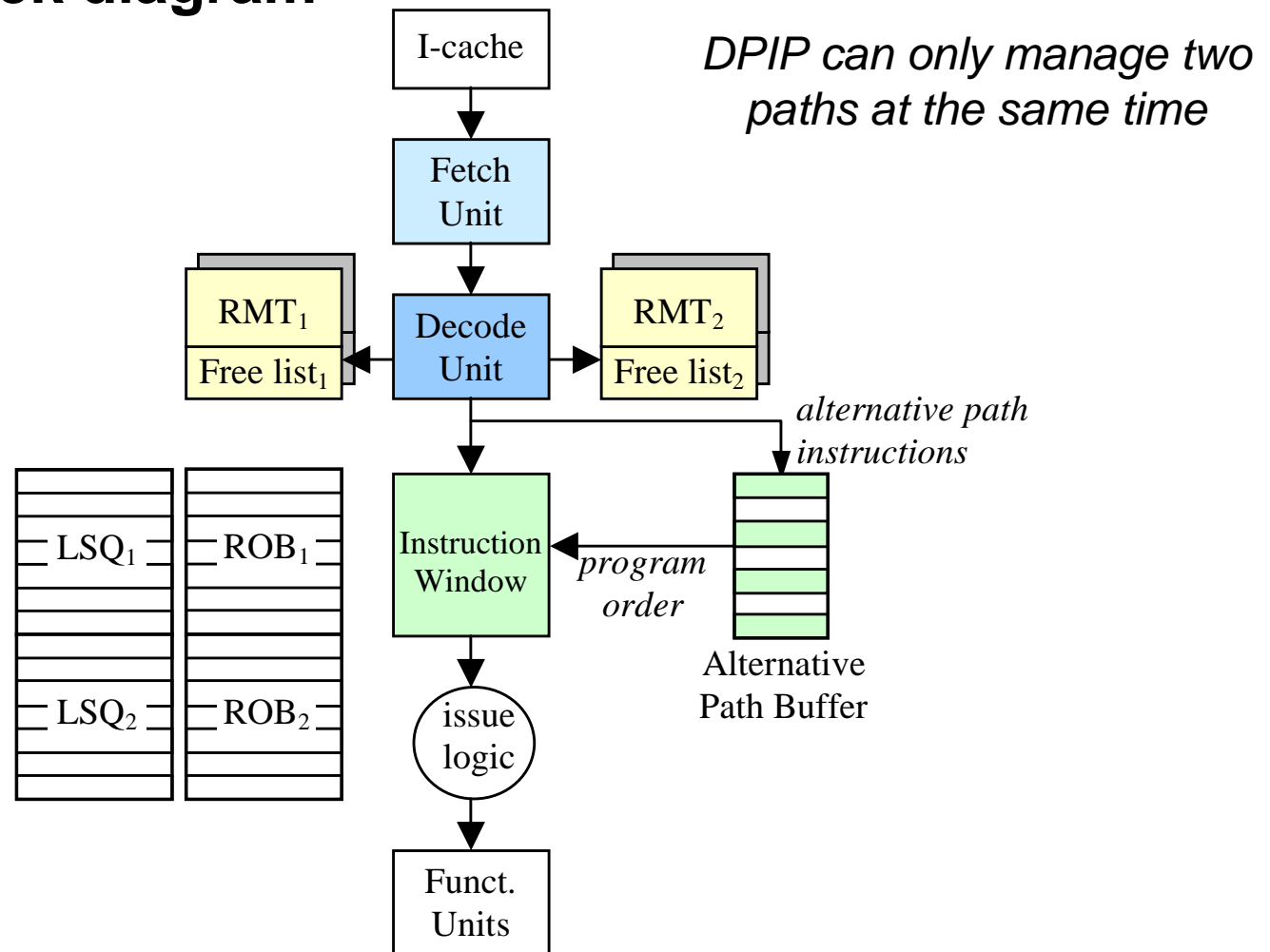
# Outline

- ▶ **Misprediction Penalty Analysis**
- ▶ **Proposal**
- ▶ **Dual Path Instruction Processing (*DPIP*)**
- ▶ **Experimental Results**
- ▶ **Sensitivity Analysis**
- ▶ **Conclusions**



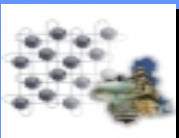
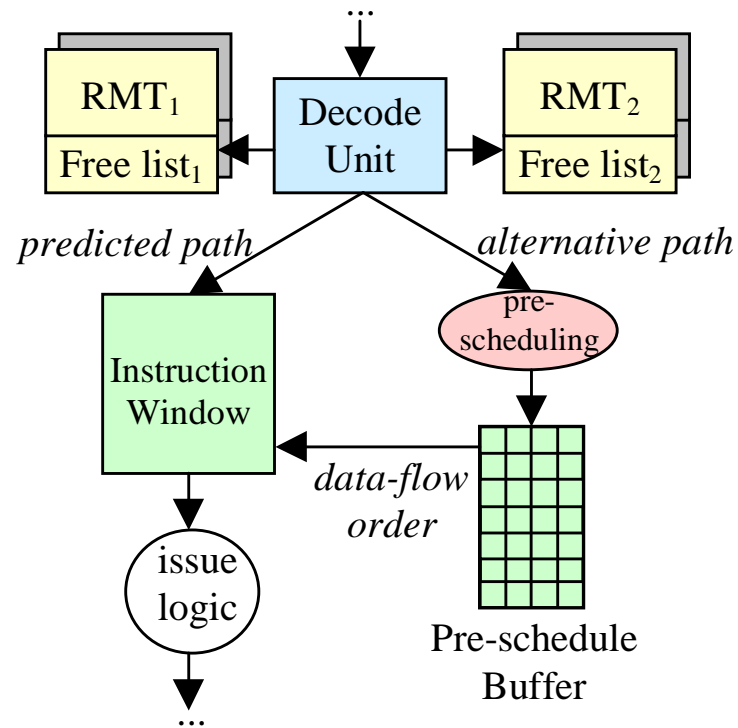
# Dual Path Instruction Processing

## ▶ *DPIP* block diagram



# DPIP

## ▶ Pre-scheduling alternative path instructions



# DPIP

## ► Pre-scheduling Example

```
schedule_line = max( {reg_availability(input reg1),  
                    reg_availability(input reg2)} )  
reg_availability(output register) = schedule_line +  
execution_latency
```

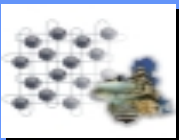
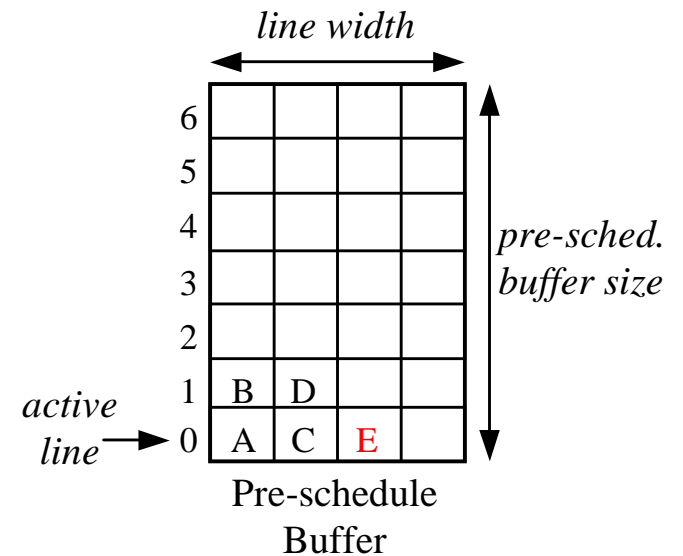
Alternative path instructions:

- A  $r2 \leftarrow r1 + r0$
- B store  $r3, 0(r2)$
- C load  $r2, 0(r6)$
- D  $r4 \leftarrow r2 + r0$
- E  $r4 \leftarrow r3 + r3$

logical register →

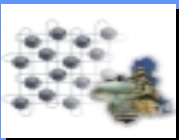
r6	0
r5	0
r4	0 2 1
r3	0
r2	0 1
r1	0
r0	0

Register Availability Table



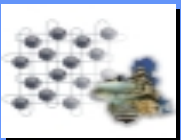
# Outline

- ▶ **Misprediction Penalty Analysis**
- ▶ **Proposal**
- ▶ **Dual Path Instruction Processing (*DPIP*)**
- ▶ **Experimental Results**
- ▶ **Sensitivity Analysis**
- ▶ **Conclusions**



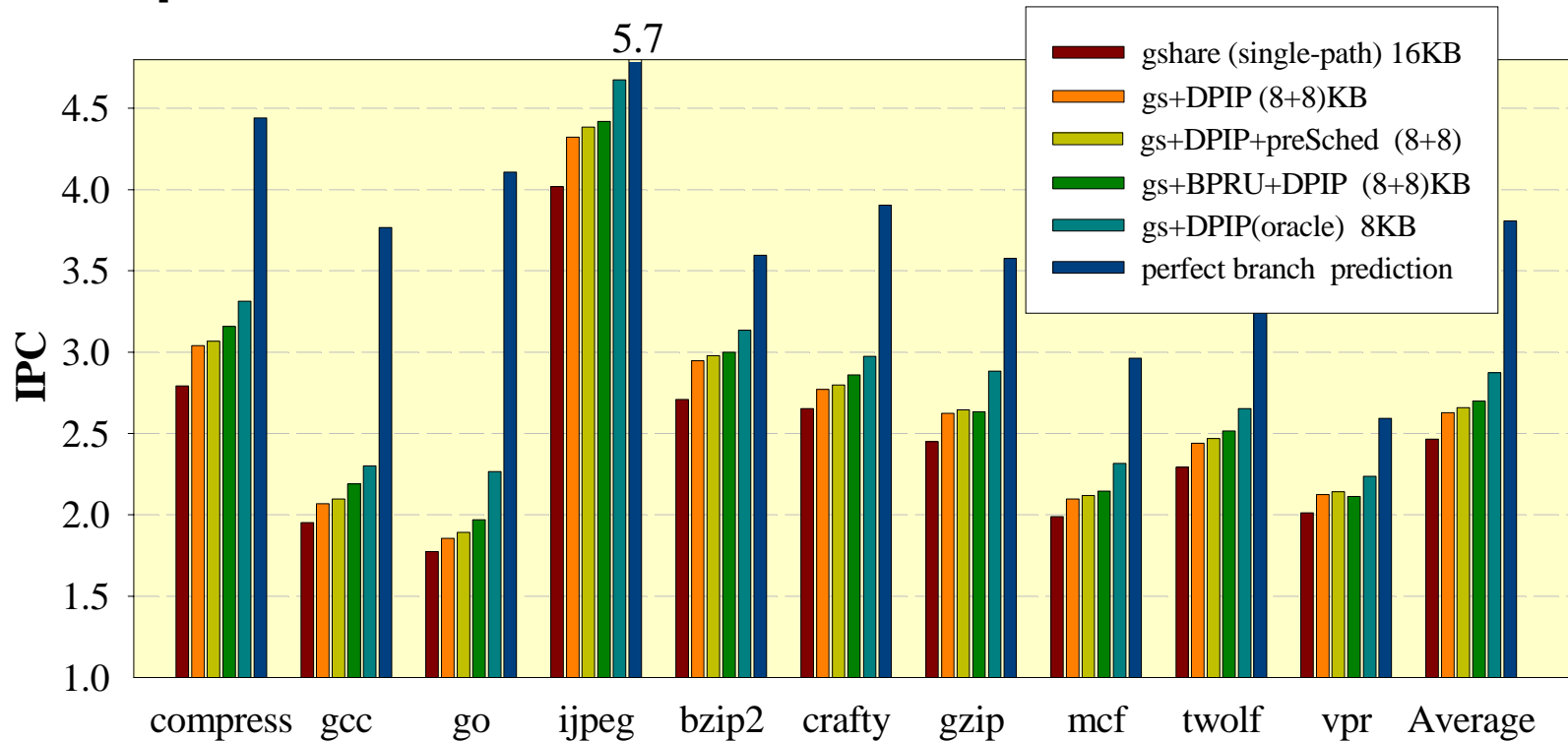
# Results

- ▶ **OoO superscalar simulator (sim-outorder)**
- ▶ **Configuration**
  - Fetch/decode/issue/commit up to 8 inst/cycle
  - L1 cache: 64 KB I-cache, 64 KB D-cache (2 way)
  - L2 cache: 512 KB 4-way
  - 8 Int ALU's, 2 Int Mult
  - 8 FP ALU's, 2 FP mult
  - 64-entry Instruction Window
  - 128-entry Reorder Buffer
  - **14-stage pipeline (IBM Power 4 - like)**
- ▶ **Evaluated programs**
  - SpecInt95 and SpecInt2000

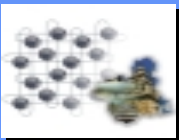


# Results

## ▶ *DPIP* performance



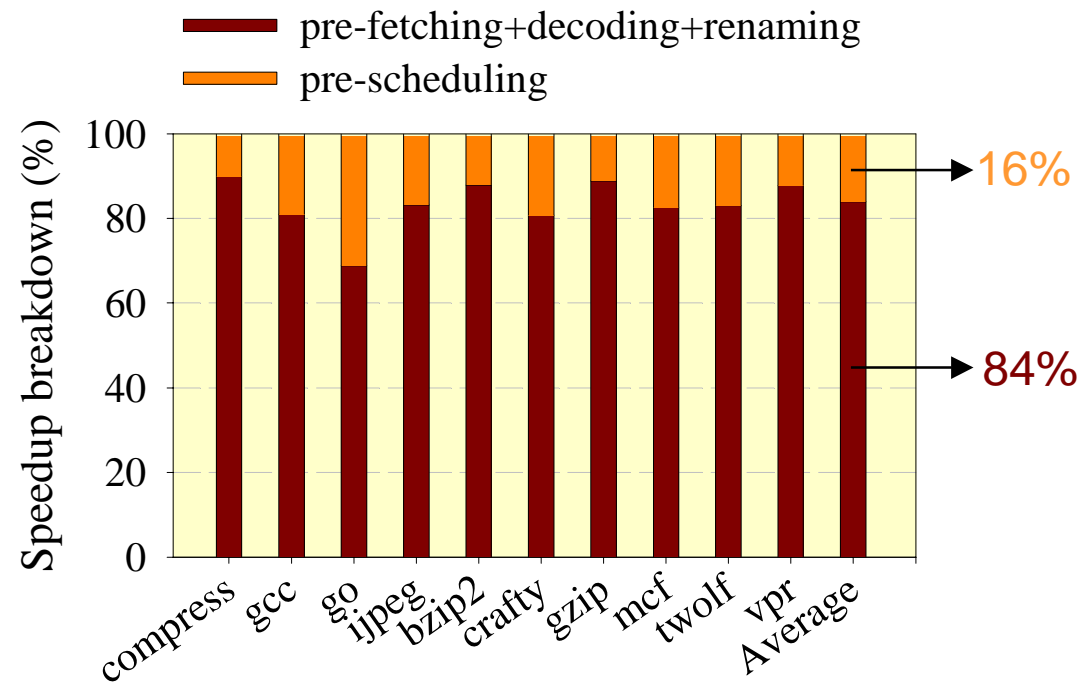
- **8%** improvement for *DPIP* (with pre-scheduling)
- **10%** improvement for *DPIP* + branch prediction reversal
- **17%** for oracle estimation (still work to be done)



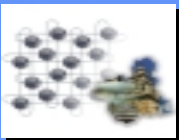


# Results

## ▶ How much pre-scheduling influences *DPIP* performance?

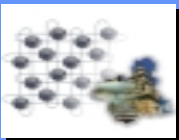


- **16%** of improvement provided by pre-scheduling (**31%** for *go*)
- Pre-scheduling provides additional benefits.



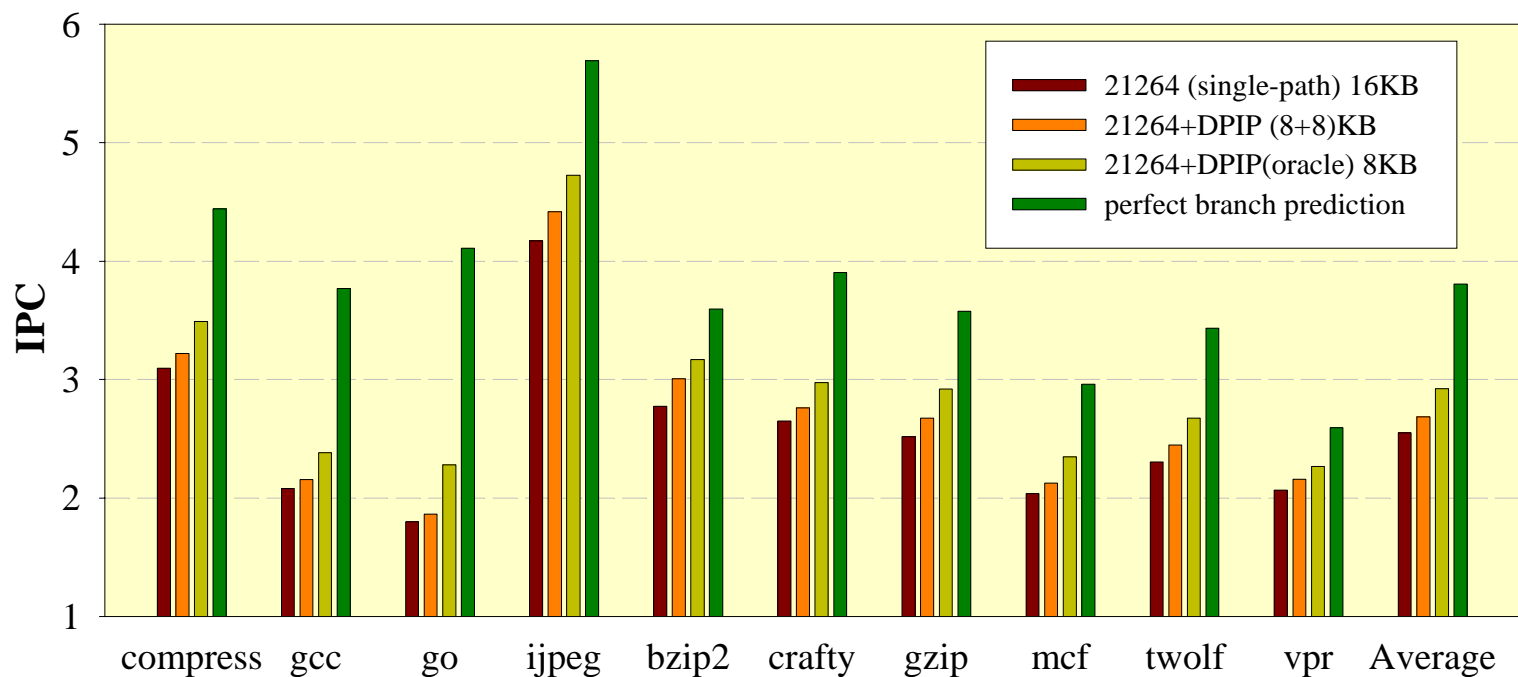
# Outline

- ▶ **Misprediction Penalty Analysis**
- ▶ **Proposal**
- ▶ **Dual Path Instruction Processing (*DPIP*)**
- ▶ **Experimental Results**
- ▶ **Sensitivity Analysis**
- ▶ **Conclusions**



# Sensitivity Study

## ▶ Alpha 21264 branch predictor

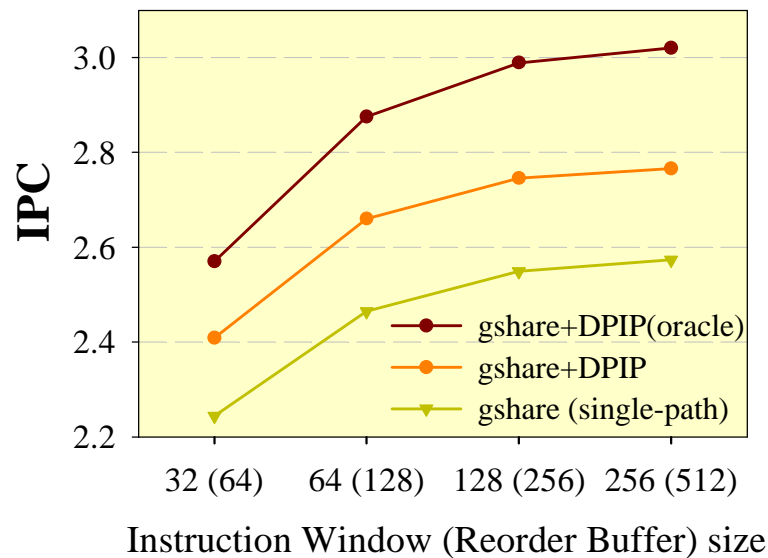


- **5%** average speedup (up to **8%** for *bzip2*)
- **15%** for oracle estimation

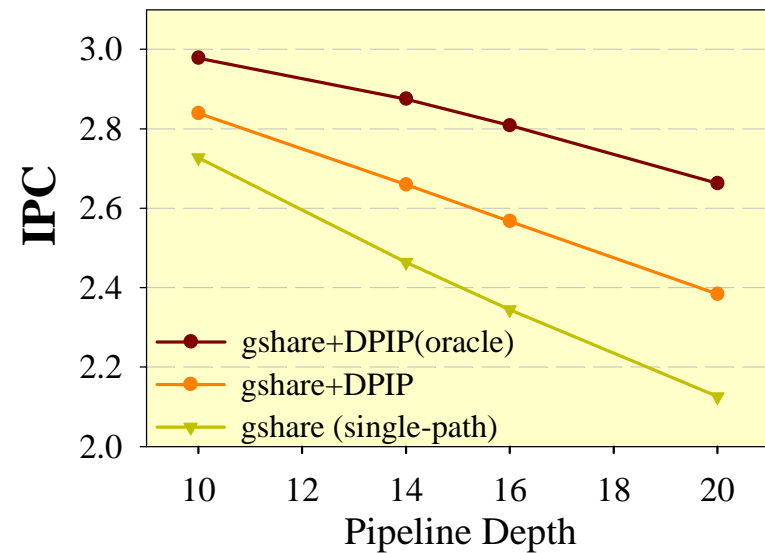


# Sensitivity Study

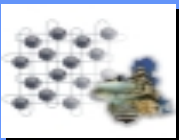
## I-Window size



## Pipeline Depth



- Improvements remain constant as the window grows
- 10 stages: **4%** average speedup
- 20 stages: **12%** average speedup



# Outline

- ▶ **Misprediction Penalty Analysis**
- ▶ **Proposal**
- ▶ **Dual Path Instruction Processing (*DPIP*)**
- ▶ **Experimental Results**
- ▶ **Sensitivity Analysis**
- ▶ **Conclusions**



# Conclusions

## ▶ Categorized branch misprediction penalty

- Pipeline-fill penalty
  - Window-fill penalty
  - Serialization penalty
- } → *contribution to the overall performance loss*

## ▶ Dual Path Instruction Processing reduces penalties of mispredicted branches

- Fetches, decodes, renames and pre-schedules alternative path instructions
- Similar effect as filling the window completely

## ▶ Balance between complexity/cost/performance

- Simpler than Multiple Path Execution schemes

## ▶ 12% speedup for 20-stages OoO processors

- 25% for oracle estimation

