

# Performance Bug-Free CPU Modeling Using Accurate Timing Simulation

Hanul Roh Dongju Chae Jangwoo Kim  
High Performance Computing Lab., POSTECH

## Motivation

- How to develop an accurate timing simulator to model modern CPUs?
- How to validate your simulation results match your design spec?

## Summary

- We combine stall-based CPI stacking and critical path analysis.
- Our approach identifies performance bottlenecks and their impacts more accurately than existing methods (FMT [1]).
- Our CPI stack is similar with ideal CPI breakdown.

## Background

- CPI : Cycles Per Instructions
  - > how many cycles to execute one instruction?
- FMT (interval analysis [1][2])
  - > CPI stack:  $\sum$  (CPI components per uArch stall event)
  - > how many cycles lost due to a specific stall event? (e.g., cache miss, branch misprediction, slow execution)

**However, inaccurate analysis due to overestimation**

## Our Idea: Avoid Overestimation

- Examine 'tagged' instructions to see exact pipeline timing.

	afetch	fetch	dispatch	ready	issue	complete	commit	info[3]	dep[3]
inst i-1	104	104	111	120	120	121	125	0 0 0	2 30 30
inst i	104	113	120	121	121	122	126	1 0 2	30 0 0
inst i+1	124	374	381	381	383	385	389	5 0 0	0 2 29

Figure 2. Example Trace Format with Timing Tags

## Further Improvement #1

- Apply Critical Path Analysis [3] to explain stall events better!
- Analyze inter-instruction dependencies.

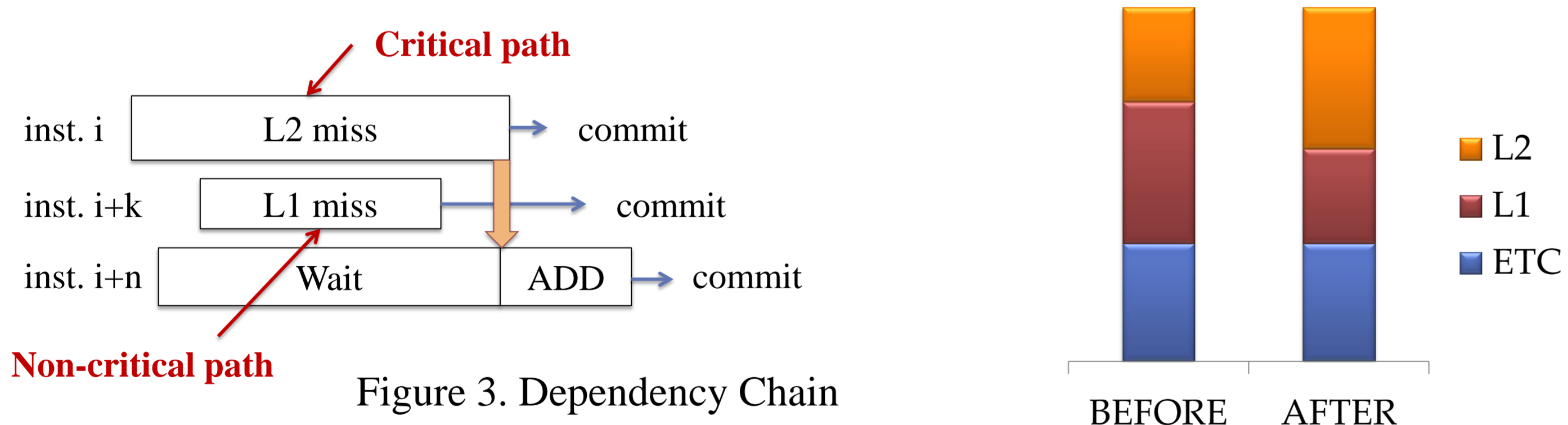


Figure 3. Dependency Chain

## Further Improvement #2

- if we removed an critical-path event?
  - > a hidden 'new' critical path appears.
- Analyze the impact of hidden critical paths.

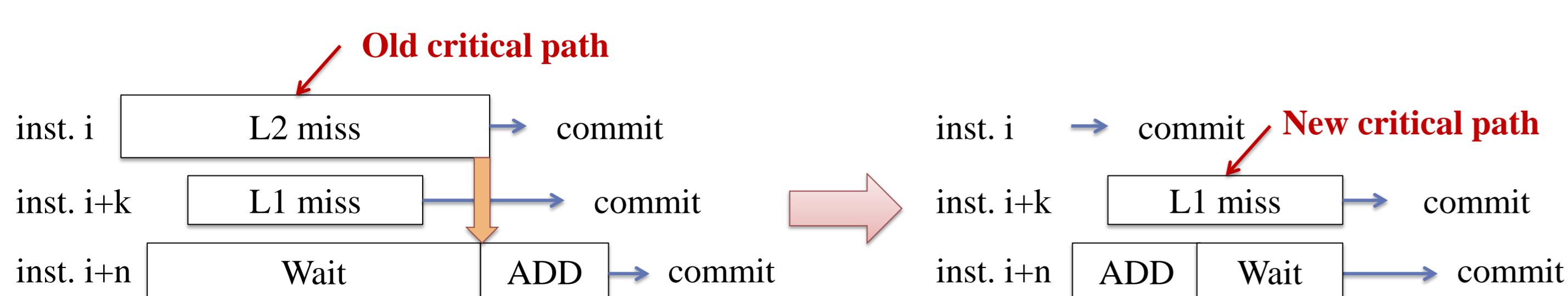


Figure 4. Impact of Hidden Critical Path

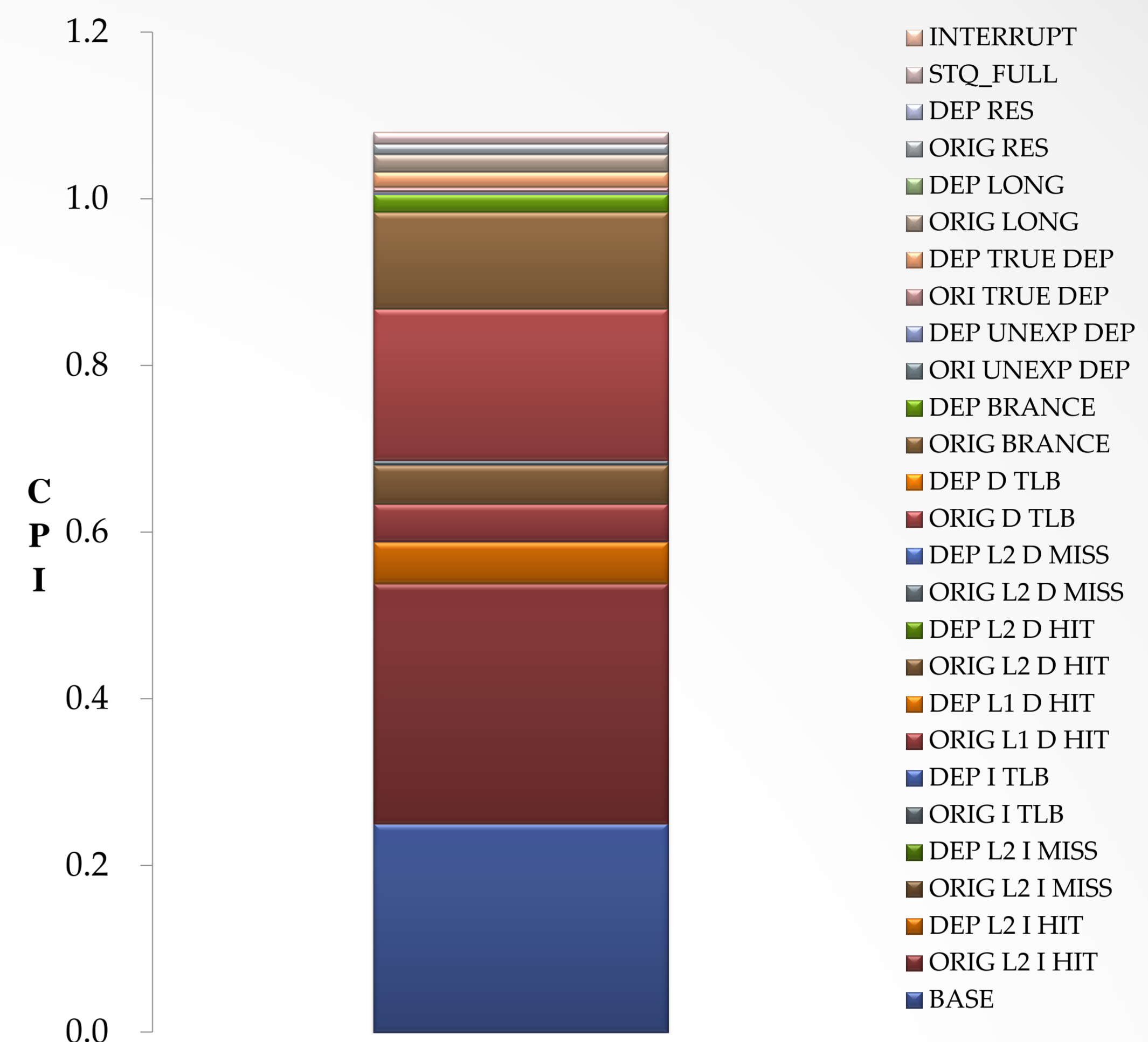
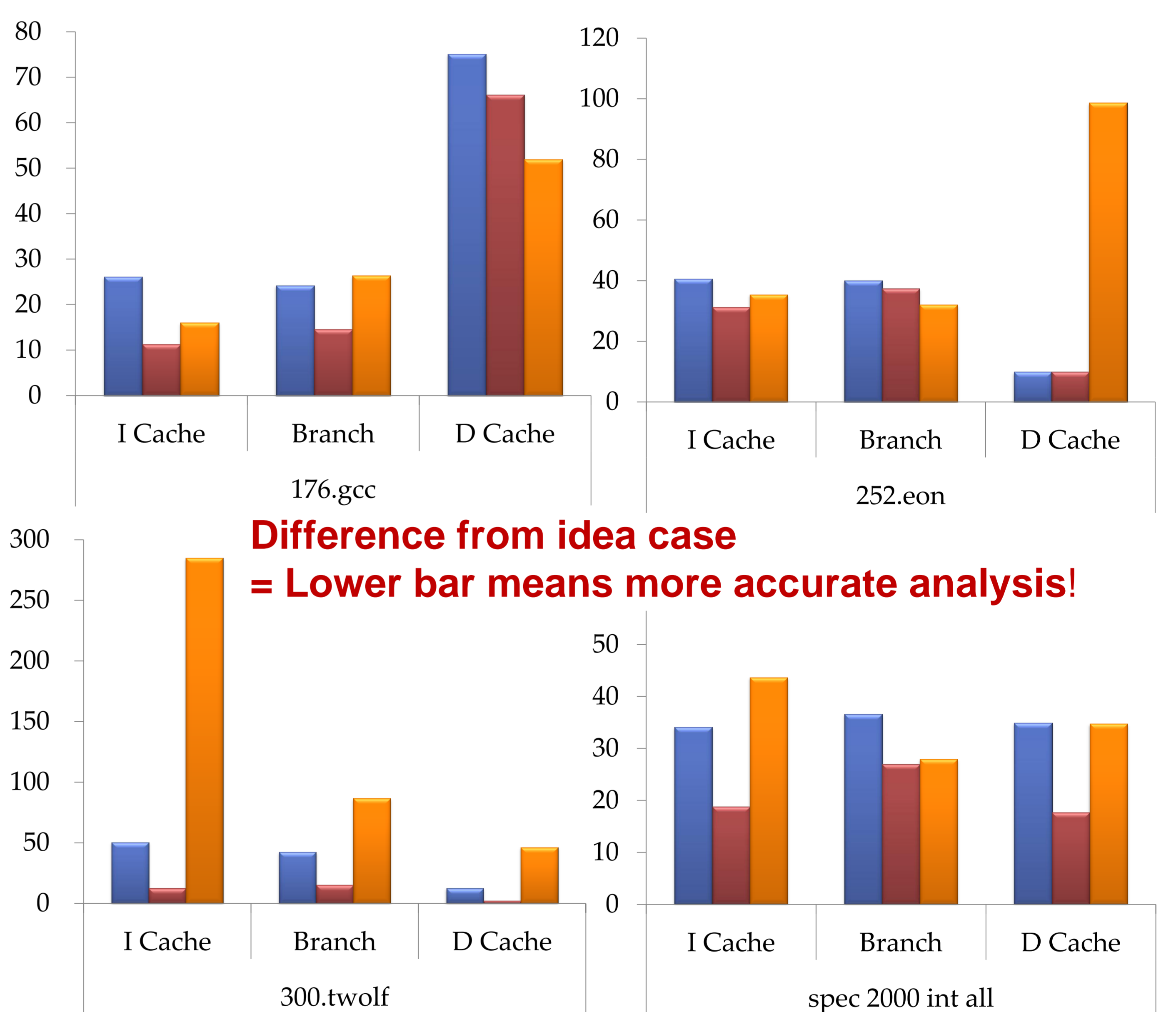


Figure 1. CPI Stack (164.gzip)

## Experimental Results (y-axis %)



Workload	Miss Events	Inst. Cache/TLB Miss	Branch Misprediction	Data Cache/TLB Miss
176.gcc		17% → 11%	27% → 14%	52% → 66%
252.eon		36% → 31%	33% → 37%	99% → 9%
300.twolf		284% → 12%	86% → 15%	46% → 2%
SPEC 2000 int all benchmarks		44% → 18%	28% → 26%	35% → 17%

Table 1. Improvement of CPI Accuracy

## Conclusion

- We construct more accurate CPI analysis methods.
- Our critical-path aware methods outperform existing methods significantly.

## References

- [1] S. Eyerman, et al, "A performance counter architecture for computing accurate CPI components" ACM international Conference on Architectural Support for Programming Languages and operating Systems, 2006, p175-184
- [2] S. Eyerman, et al, "A Mechanistic Performance Model for Superscalar Out-of-Order Processors" ACM Transactions on Computer Systems, 2009, p3:1~3:36
- [3] Brian Fields, et al, "Focusing Processor Policies via Critical-Path Prediction" ISCA, 2001, p1~12