Intelligent wrong path issue: a means of improving performance

By

Chidiogo Madubike

Meghana Sardesai

Nmita Sarna

Introduction

- Speculation of instructions is crucial for modern superscalars
- Branch predictors provide accuracies of up to 96% and are key to effective speculation
- However we cannot rely solely on prediction because the latency of misprediction has a negative impact on performance
- To solve this problem, computer architects came up with the idea of wrong path issue

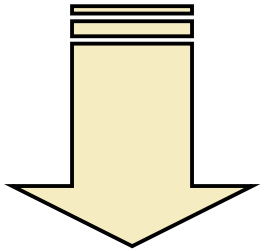
To issue or not to issue?

- Wrong path issue reduces the misprediction penalty
- On the other hand, useless wrong path instructions waste processor resources and must invariably be flushed from the pipeline
  - Have to have enough resources to issue and execute 2 paths knowing that one of them will be invalidated
- To find a better solution, we look at the behavior of branch instructions:

Branch behavior

- Program trace- branch instructions

  1 0 1 1 1 1 1 1 1 0 0 1 1 0 1 1 0 0 1 0 1 1 0 1 0 1 1 1 1 1 1 0 1
  1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1



- *BLEZ instruction.*

- **……..01101110**11111111111111111111**……..**

Idea

- Improve performance by introducing an algorithm which only selectively issues from the wrong path by observing the behavior of individual branches within a program

- Attempts to:

  - Reduce number of useless instructions being executed thus increasing useful instruction throughput
  - Reduce complexity involved in always issuing and executing of wrong path instructions
  - Prevent flushing of these instructions whenever the prediction is correct and therefore improving usage of resources and performance

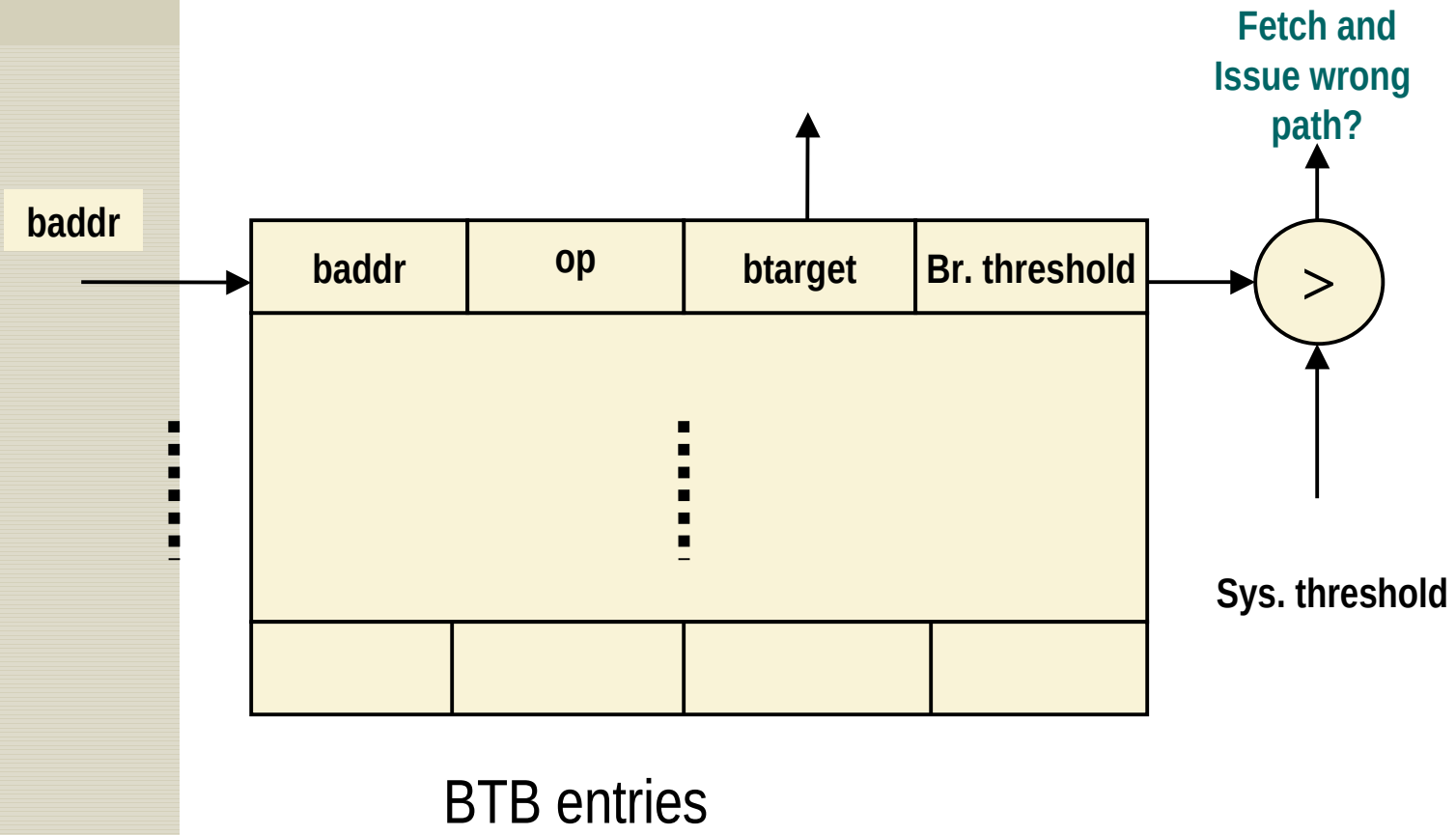
Resource Usage

- # of functional units, dispatch queues, IFQs, reservation stations, etc.
  - For programs without enough ILP to make full use of processor resources, this might not be a problem
  - However if we have any programs that can make sufficient use of the processor resources then something needs to be done
- Attempt to improve resource usage using a new algorithm

Implementation details

- Branch address sent to predictor
- Predictor entry for each branch includes
  - Threshold value of particular branch
- Predictor includes confidence values which contain
  - *Increment* value for each correct prediction
  - *System threshold* value – (have to determine optimum value)
  - *Max* value of branch threshold
  - *Prediction penalty* for wrong predictions

Architecture



Program trace:

- Our program traces indicate that we attempt to capture those areas of the program in which the branch behavior is predicted to be highly accurate

:

op = 6 ; 0 br thr= 21 path dec= 1

op = 2 ; 1 br thr= 27 path dec= 1

op = 3 ; 1 br thr= 28 path dec= 1

op = 6 ; 0 br thr= 17 path dec= 1

op = 2 ; 1 br thr= 28 path dec= 1

op = 3 ; 1 br thr= 29 path dec= 0

op = 6 ; 0 br thr= 25 path dec= 1

op = 6 ; 0 br thr= 13 path dec= 1

op = 6 ; 1 br thr= 14 path dec= 1

:

Simulations

- We performed our simulations on spec2000 reduced benchmarks using two different #'s of execution resources
- Our benchmarks : equake (FP), mcf (INT), parser (INT)
  - Equake: Simulation of seismic wave propagation in large basins
  - Mcf: Combinatorial optimization / Single-depot vehicle scheduling
  - Parser: Word Processing
- The programs we simulated didn't have much parallelism but we were able to obtain data that proved our hypothesis to some degree

Simulations cont'd…

- 1st set of data
  - 4 FPU adders
  - 4 integer ALUs
  - 1 integer MULT/DIV unit
  - 1 FPU MULT/DIV unit
- 2nd set of data
  - 3 FPU adders
  - 3 integer ALUs
  - 1 integer MULT/DIV unit
  - 1 FPU MULT/DIV unit

- Baseline Architecture – Always issues and executes the wrong path
- Our program – Selectively issues and executes the wrong path depending on information from the branch predictor

Results for FU = 4

- Baseline

Benchm.	CPI	IPC	Exec_BW
Equake	0.7175	1.3938	1.5320
Mcf	1.4201	0.7042	0.7672
Parser	0.6121	1.6338	1.9959

- Our Dynamic Program

Benchm.	CPI	IPC	Exec_BW
Equake	0.7181	1.3927	1.4303
Mcf	1.4250	0.7018	0.7466
Parser	0.6036	1.6567	1.7259

Equake = 0.11% degradation (IPC)
Mcf = 0.24% degradation (IPC)
Parser= 2.29% improvement (IPC)

Results for FU = 3

- Baseline

Benchm.	CPI	IPC	Exec_BW
Equake	0.7225	1.3840	1.5256
Mcf	1.4174	0.7055	0.7700
Parser	0.6228	1.6057	1.9657

- Our Dynamic program

Benchm.	CPI	IPC	Exec_BW
Equake	0.7231	1.3829	1.4204
Mcf	1.4271	0.7007	0.7450
Parser	0.6146	1.6272	1.6877

Equake = 0.11% degradation (IPC)
Mcf = 0.48% degradation (IPC)
Parser= 2.15% improvement (IPC)

Conclusions

- Reduction of useless resource usage
- Effect of ILP
- Effect of more or less execution resources
- Effect on longer pipelines
- Validity of Hypothesis

References

- Swanson, McDowell, Swift, Eggers, Levy. "An evaluation of Speculative Instruction Execution on Simultaneous Multithreaded processors", submitted for publication.
- Johnny K.F. Lee, Alan Jay Smith, "Branch Prediction Strategies and Branch Target Buffer Design", Computer, January 1984.
- Joel Emer, Nikolas Gloy. "A Language for Describing Predictors and its Application to Automatic Synthesis", Proceedings of the 24th International Symposium on Computer Architecture, June 1997
- Eric Rotenberg, Steve Bennet, James E. Smith, "Trace Cache: a Low Latency Approach to High Bandwidth Instruction Fetching", Proceedings of the 29th International Symposium on Microarchitecture, December 1996.
- Daniel Holmes Friendly, Sanjay Jeram Patel, Yale N. Patt, "Alternative Fetch and Issue Policies for the Trace Cache Fetch Mechanism", Proceedings of the 30th International Symposium on Microarchitecture, November 1997.