Towards an Energy Efficient Branch Prediction Scheme Using Profiling, Adaptive Bias Measurement and Delay Region Scheduling (IEEE DTIS '07)

M. A. HICKS*, C. EGAN, B. CHRISTIANSON AND P. QUICK
Compiler Technology and Computer Architecture Research Group
University of Hertfordshire
*m.hicks@herts.ac.uk

ABSTRACT

Dynamic branch predictors can account for over 10% of a processor's power consumption. This power cost is proportional to the number of accesses made to that dynamic predictor during a program's execution. In this work we propose the combined use of local delay region scheduling and profiling with an original adaptive branch bias measurement. Our adaptive branch bias measurement takes note of the dynamic predictor's accuracy for a given branch and decides whether or not to assign a static prediction for that branch. The static prediction and local delay region scheduling information is represented as two hint bits in branch instructions. We show that, with the combined use of these two methods, the number of dynamic branch predictor accesses/updates can be reduced by up to 62%. The associated average power saving is very encouraging; for the example high-performance embedded architecture n average global processor power saving of 6.22% is achieved.

1 Introduction

The latency associated with branch instructions can be overcome by various means. Currently, state-of-the-art processors tend to use dynamic branch prediction, but the use of dynamic predictors can consume large amounts of the power budget. In current processors, a branch predictor can consume over 10% of the overall CPU power budget. The power cost is directly proportional to the number of accesses made to the dynamic predictor.

However, even though a dynamic predictor uses a great deal of power, the increased prediction accuracy and improved processor performance it provides results in power saving by the reduced number of branch mispredictions, negating the necessity of stalling the processor. In this work, we present an approach for reducing the number of accesses and updates made to a dynamic branch predictor that combines scheduling the local delay region with profiling using an adaptive bias measurement. Encouraging experimental results are shown.

2 Biased Branches

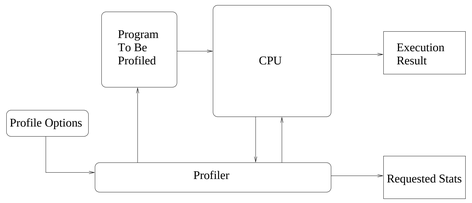
In many cases the direction of a branch tends to be biased to either the taken path or to the not-taken path and therefore demonstrates a skewed distribution. In this work, we use profiled branch prediction in conjunction with a dynamic predictor. The idea of profiled branch prediction is to avoid accessing the dynamic predictor whenever possible, thereby saving power. However, the profiled static prediction must be accurate to ensure that there is no impact on dynamic prediction accuracy, since inaccurate predictions are expensive in terms of both performance and power.

In the static code the number of biased branches appears to be small, but during program execution biased branches tend to be executed repeatedly and are therefore executed frequently.

Previous approaches have associated static predictions with biased branches by using a fixed biased level to decide whether a branch is biased or not. This is far less accurate than a typical dynamic branch prediction. In our approach we take such problems into account, and we propose the use of the adaptive branch bias measurement technique through profiling and local delay region scheduling.

3 Profiling

Profiling, in this case, refers to the observation of a given program, at the assembly/machine level, while undergoing execution with a sample dataset.This means each branch instruction can be monitored in the form of a program trace by a detailed history of selected instructions and any relevant information extracted and used to form profiled static predictions where possible. A profiler is any application/system which can produce such data by observing a running program. The number of datasets that any given program is profiled with will affect the likely 'real' accuracy of the profiling results. Profiled traces permit the exact bias of a branch instruction to be known resulting in higher prediction accuracies.



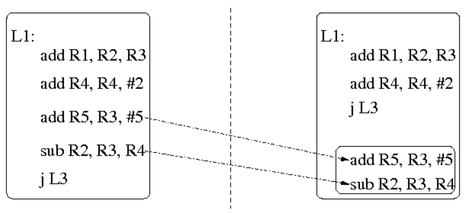
Example of Profiling

3.1 Adaptive Bias Measurement

Profiled static hints will be out performed by dynamic prediction for many branches unless the hinting bias level is set so high that only extremely biased branches are removed and therefore profiling should be used with due caution. Consequently, we only only assign a profiled prediction to a branch where avoiding dynamic prediction has no significant negative impact on that branch's dynamic prediction accuracy. When profiling each branch in a program's execution, our profiler records the directional history for each branch, and also the prediction history. From this record or trace, we compute whether a branch's bias is equal to, or greater than its associated prediction accuracy from the dynamic predictor. This computation is key to the results we present in this work. Assuming a program is profiled against an adequately varied data set, we show that these branches can safely be removed from dynamic prediction through the use of profiled hint bits.

4 Local Delay Region Scheduling

Local delay region scheduling is the process of scheduling branch independent instructions from before the branch in the same basic-block into the branch delay slots to be executed by the processor after the branch. A branch independent instruction is any instruction whose result is not directly, or indirectly, depended upon by the branch to compute its own behaviour. The locally scheduled delay region, for a given branch, is executed irrespective of the branch direction outcome, and removes the need to predict for any branch where it can be used.

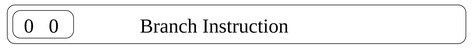


Simple Local Delay Region Scheduling (2 Delay Slots)

5 Hardware Implementation/Hint-Bits

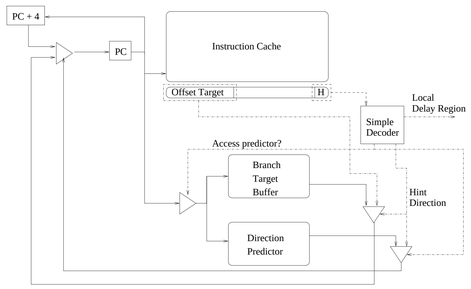
The hardware requires support for the following behaviours:

1. Statically predict taken. Do not access, or update, the dynamic predictor for this branch.
2. Statically predict not-taken. Do not access, or update, the dynamic predictor for this branch.
3. Use the locally scheduled delay region. Do not access, or update, the dynamic predictor for this branch.
4. Use the dynamic predictor.



Hint-Bits in Branch Instruction Format

These behaviours are represented using two additional bits in branch instructions (hint-bits), but with a special power saving implementation that avoids accessing/updating the dynamic branch predictor for those branches that are assigned hints. These modifications are in the Instruction Fetch and EXEcution pipeline stages.



Logic Modifications Required in IF Pipeline Stage

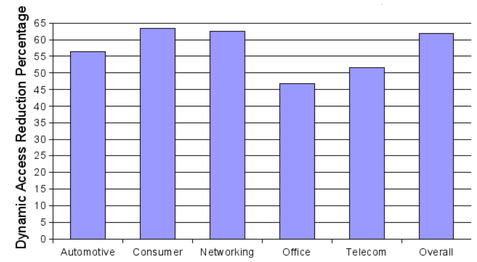
6 Scheduling & Hinting Algorithm

Input: All Assembly Files of Programs
Output: Appropriately Hinted Assembly Files
foreach *Program* **do**
  foreach *Assembly File* **do**
    foreach *Branch Instruction* **do**
      Initially, set hint bits to "Use the dynamic predictor for this branch"
      if *Can Schedule Local Delay Region* **then**
        Set hint bits to use local delay region and move two instructions preceding branch into delay region (if possible)
      else
        if *Branch's Profiled Bias* $\geq$ *Dynamic Branch Predictor's Accuracy for this Branch* **then**
          Set hint bits to Predict Profiled Bias
        end
      end
    end
  end
end

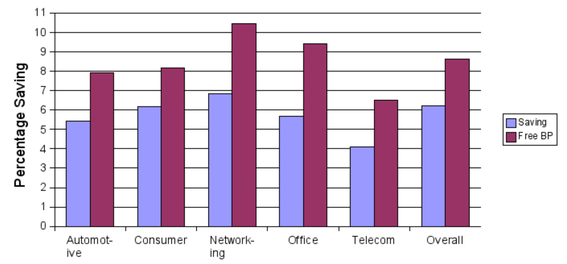
7 Experimentation

The proposed 'Combined Algorithm' was applied to the EEMBC benchmark suite. The performance was then evaluated using a modified version of the Wattch power analysis processor simulator (itself a version of SimpleScalar). **Baseline Configuration: 2-Way Issue, 3 Delay Slots, GShare Branch Predictor, 512 Entry BTB, Non-Ideal Conditional Clocking, 'GCC -O2'.**

7.1 Results



Percentage of Accesses/Updates of Branch Predictor Avoided



Global Processor Energy Saving Per Committed Instruction (%)

8 Conclusions

- Around 63% of dynamic branch predictor accesses can be prevented using this algorithm
- On the experimental baseline this results in a global power saving of over 6%
- Branch prediction accuracy is not decreased (in fact, in many cases it is improved)
- Minimal hardware modifications are required

References

[1] Parikh, D., Skadron, K., Zhang, Y., Stan, M.: Power aware branch prediction: Characterization and design. IEEE Transactions On Computers 53(2) (February 2004)

[2] Hicks, M., Egan, C., Christianson, B., Quick, P.: Towards an energy efficent branch prediction scheme using profiling, adaptive bias measurement and delay region scheduling. In: Design and Technology of Integrated Systems, IEEE (September 2007)