Gshare, "Agrees" Aid Branch Prediction

New Algorithms Have Minimal Hardware Cost But Improve Accuracy

by Linley Gwennap

Several recently announced microprocessors have adopted one (or both) of two new algorithms for branch prediction. The first, known as Gshare, is a method of indexing into a large branch history table (BHT). The second, "agrees" mode, is a method of encoding the history bits themselves. Although both methods offer modest improvements in branch-prediction accuracy, neither has significant hardware cost compared with current methods. With modern microprocessors facing lengthier branch penalties, both techniques are likely to become standard practice as designers revise their processors over the next couple of years.

Gshare Improves Two-Level Algorithm

The Gshare algorithm is a variation of the two-level algorithms first published by Yeh and Patt (see MPR 3/27/95, p. 17). The complete two-level algorithm uses a table of branch patterns to index into a table of history values. Because this algorithm requires a time-consuming pair of lookups, commercial processors have generally used a simplified version in which a global history value is used to index into the history table.

To allow some amount of per-branch history, this global history is typically concatenated with some bits of the branch address. A paper by Digital's Scott McFarling (*www.research.digital.com/wrl/techreports/abstracts/TN-36.html*) calls this method Gselect. It improves performance over an index that ignores global branch history due to the correlations between nearby branches. For example, in the sequence

```
IF (x<1)...
IF (x>1)...
```

the direction of the second branch is clearly influenced by the outcome of the first branch.

For a BHT with a fixed number of entries e, the size of the index is $\log_2(e)$ bits. For example, a 1,024-entry BHT requires a 10-bit index. Using Gselect, these bits must be divided between the global history and the branch address, for example, 5 bits of each.

McFarling notes that the indexes generated by Gselect have a lot of redundancy, since few combinations are relevant. Thus, hashing the global history and the branch address using an XOR operation will typically not remove useful information. Prediction is improved because more bits of the global history and the branch address can be used. In the 1,024-entry BHT, up to 10 bits of each could be combined to create the index. McFarling calls this method Gshare.

Extending the global branch history beyond several bits does not significantly improve accuracy and can instead introduce noise into the hashed index. Thus, for a large BHT, the index may use only 4–8 bits of global history XORed with the upper bits of the branch address.

The hardware cost of implementing Gshare instead of Gselect is negligible: possibly extending the size of the global branch history register by a few bits and adding a set of XOR gates to the index logic. According to McFarling, Gshare improves accuracy by less than 1% over Gselect, but the cost is so small that there is no reason not to use the better method.

Agrees Mode Avoids Contention

A typical BHT uses two bits to encode the history of a particular branch (see MPR 3/27/95, p. 17). These bits indicate if the branch has been taken or not taken. The PA-8500 (see MPR 11/17/97, p. 20) uses a different encoding: the bits indicate whether the outcome of the branch has agreed or disagreed with the static prediction. In PA-RISC, branches include a static prediction bit in the instruction itself.

At first glance, there appears to be no practical difference between the two methods. Because most BHTs have no tags, however, multiple branches might map to the same BHT entry. Sometimes, a new branch simply takes over an entry from an old branch. In the worst case, two active branches can both be updating a single BHT entry.

In the taken/not taken scheme, there is a 50% chance that the two conflicting branches will go in opposite directions, causing frequent mispredictions. If both branches are following their static predictor, however, the agrees method will correctly predict both branches even though they map to the same BHT entry. The chances of a misprediction are reduced to $2p(1–p)$, where p is the success rate of the static predictor. For p=0.7, the agrees mode mispredicts only 42% of the time in situations where two branches map to the same BHT entry. While this improvement is small, the hardware cost of implementing agrees mode is essentially zero: since only the encoding is changed, no new storage is needed.

Agrees mode is an obvious solution for instruction sets —like PA-RISC, PowerPC, and MIPS—that include static branch prediction. Centaur found a way to make it work with x86, which does not. The C6+ (see MPR 11/17/97, p. 17) generates a static prediction based on the opcode and direction (forward or backward) of the branch, then compares this prediction with the actual result to encode the branch history using agrees mode. This on-the-fly prediction adds hardware cost, but static prediction can be done very simply (the simplest way is backward taken, forward not taken), allowing agrees mode to be applied to any instruction set. Thus, like Gshare, agrees mode is likely to become widely used. M