Cache Design with Path Balancing Table, Skewing and Indirect Tags

Tommi Jokinen and Chia-Jiu Wang

University of Colorado at Colorado Springs
Department of Electrical and Computer Engineering
P.O. Box 7150, Colorado Springs, CO 80933 - 7150

Abstract

A cache design aimed to reduce cycle time, area, and miss rate simultaneously is proposed in this paper. A cache with path balancing table, skewing, and indirect tags can reduce cycle time, use less area, and reduce miss rate at the same time as compared with a baseline cache. From our simulation results, we propose cache design alternatives in achieving lower cycle time, lower miss rate, and using less area.

1. Introduction

In this paper, we present a cache design with techniques to reduce cycle time, save area, and reduce miss rate. We first review the techniques briefly, then we present our cache design by combining unique features of different techniques to achieve an improved high performance cache architecture.

Seznec [1], introduces the *indirect-tagged* cache to reduce area cost. The idea behind this method is to recognize there are actually three places where the page numbers are stored inside a microprocessor. These three places are Translation Look-aside Buffer (TLB), cache tag, and branch target buffer A pointer or an *indirect tag* to a location where the page number is stored is used instead of using address tags. This location is referred to as a *page-number cache* or a *PN-cache*. The tag array size is independent of the address width because a tag of any arbitrary size can be represented by a pointer. The size of the pointer depends on the size of the PN-cache. It is often desirable to have small cache block sizes with out-of-order execution and pipelined Level 2 caches. This goal can be achieved at little cost using this method.

Skewed-associative cache design is presented in [6],[9]. With a two-way skewed-associative cache we are able to keep the same hardware complexity as a regular two-way set-associative cache, but with the hit rate performance close to that of a four-way set-associative cache. Two-way skewed caches are a good tradeoff for small on-chip caches. The goal is to minimize hardware implementation cost and keep cache cycle time low. The performance improvement of skewed caches relies on inter-bank dispersion and not on a simple hashing algorithm.

In order to reduce the cycle time, a method called Path Balance Table (PBT) is presented in [7]. This technique helps improve performance through reducing cycle time by balancing the path delay between tag and data array. The idea is to use a separate subset of the tag array to de-couple the one-to-one relationship between the address tags and the cache blocks for set-associative caches. We call this subset of the tag array for the Path Balance Table (PBT). The path through the tag array is significantly longer than the path through the data array for both direct-mapped and set-associative caches. For set-associative caches we need to know the result of the tag comparison before we can select the correct cache block, this further lengthens the tag path. The PBT contains a subset of address tags and the corresponding way in a set to which they belong. Address tags stored in the PBT are for cache blocks recently referenced or any other replacement scheme preferred. The PBT makes the tag array independent of the data array, so the data array entry can be selected ahead of address tag comparison.

2. Proposed Cache Design

To support the discussions on combining different techniques to improve cache performance and to show that an implementation is feasible, we select one of the best combinations, and verify our design with some actual simulation results. The baseline cache organization for the simulation will be a Virtually Indexed Physically Tagged (VIPT), 2-way set-associative, Least Recently Used block-replacement, {8,16,32 and 64}-Kbytes split cache. It will be assumed that the TLB is fully-associative and with a size depending on the indirect-tag cache size, which will become apparent later. A write-through strategy is also assumed to avoid problems with page invalidation when using indirect-tags. Area cost and miss-rate will be discussed based on simulation results. Since cycle time is technology dependent, it will be based on assumptions and logical reasoning. A miss penalty of eight clock cycles for accessing data from lower memory will be assumed when needed.

We add the following features on to the baseline cache, in order, and do simulation and analysis on that organization comparing with previous organizations:

(a) Path Balancing Table (PBT) will be added as shown in Figure 1. Size will be 1:1, 1:2, 1:4 and 1:8 of the base cache size. A 1:1 ratio means that number of sets in the direct-mapped PBT cache is equal to the number of sets in the regular cache.

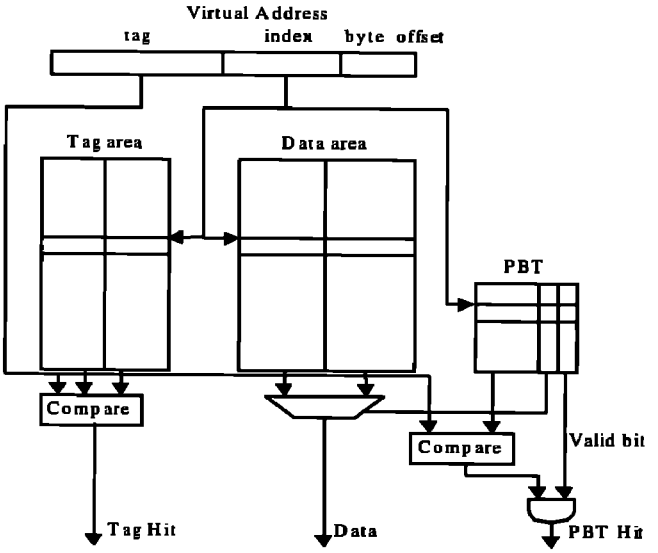

Figure 1: Baseline cache with PBT cache added. TLB not shown.

(b) Add skewing with degree eight onto cache as shown in Figure 2. Simple XOR-function will be used on the high order 3-bits of the set for instruction/data cache.

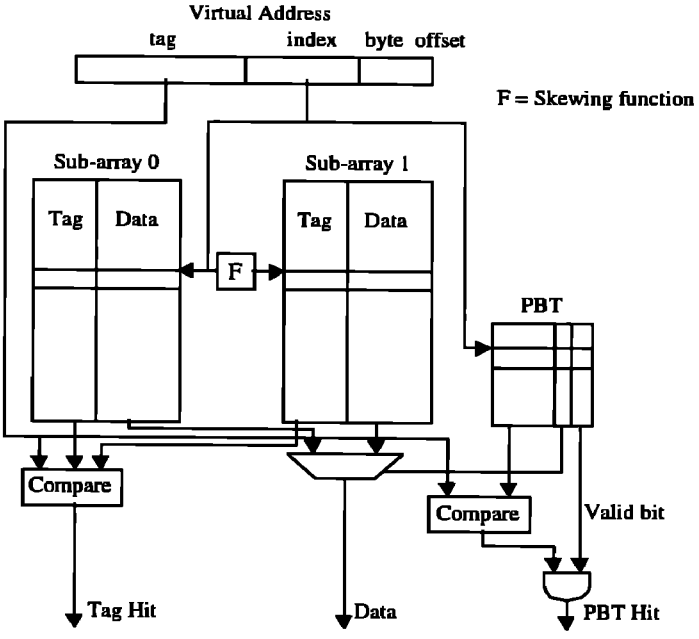

Figure 2: Baseline cache with PBT cache and skewing function added. TLB not shown.

(c) Next, we add the indirect-tags to the baseline cache as shown in Figure 3. Number of PN-cache entries simulated will be 8, 16, 32 and 64. It is assumed that the TLB size is equal or less than the PN-cache size, otherwise the TLB can not be fully utilized. In our simulation, an indirect-tagged cache will be modeled as a tag cache without a TLB. Our goal here is to be able to speed up the PBT tag path and lower the cycle time even more. We can add indirect tags to the baseline cache structure using Seznec's approach. The number of tag entries in the PN-cache will be fixed at 32. Then different PBT cache sizes will be simulated.

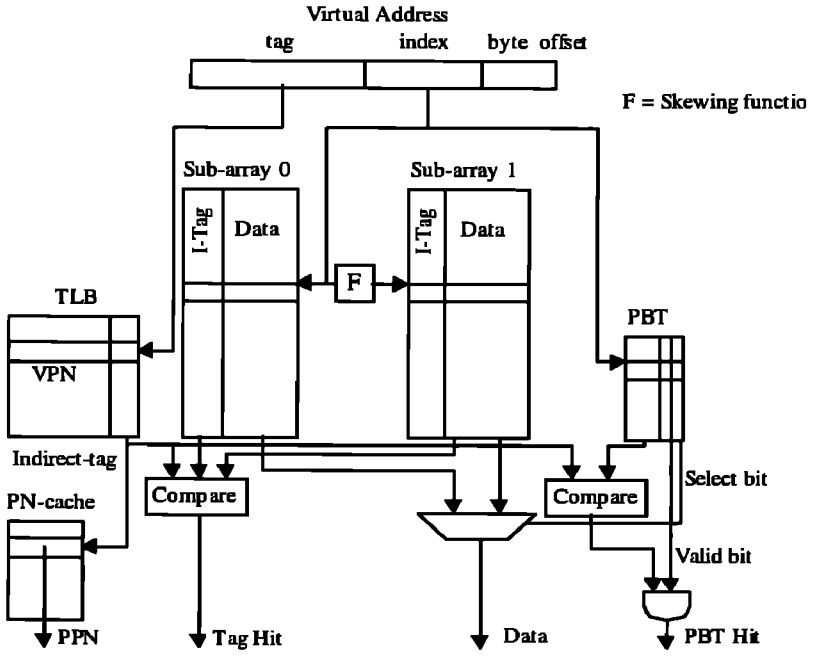

Figure 3: Baseline instruction/data cache with PBT cache, skewing and PN-cache.

A modified version of the Acme Cache Simulator from the Parallel Architecture Research Laboratory in New Mexico will be used as the simulation tool with some added features. For simulation data, we will use trace-files generated on a SPARC machine from the SPEC92 benchmark package.

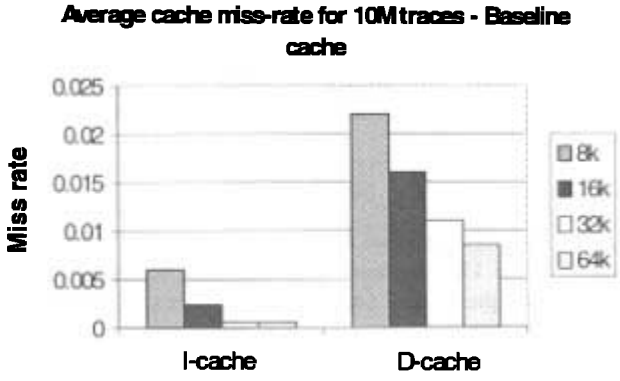

Figure 4: Average miss-rate for baseline 2-way set-associative split cache, 64 byte blocks. Data averaged from Gcc, Ear, Sc, Swm256, Li, Eqntott, Compress and Tomcatv, all 10M traces.

2.1 Cache with Path Balancing Table (PBT)

Next, we add the Path Balancing Table to the baseline cache. The PBT is a direct-mapped cache with a maximum number of entries equal to number of sets of the corresponding cache. Here we will simulate PBT cache sizes with a ratio of 1:1, 1:2, 1:4 and 1:8. We can expect the 1:1 ratio to be the best regarding miss-rate since it is the largest one. However, a large PBT is also slower, so there is a tradeoff between miss-rate and speed. For cache sizes below about 16-Kbytes, the effect of using a PBT is less beneficial because the PBT will dominate the data path [7]. Figure 5, below, supports this assumption.

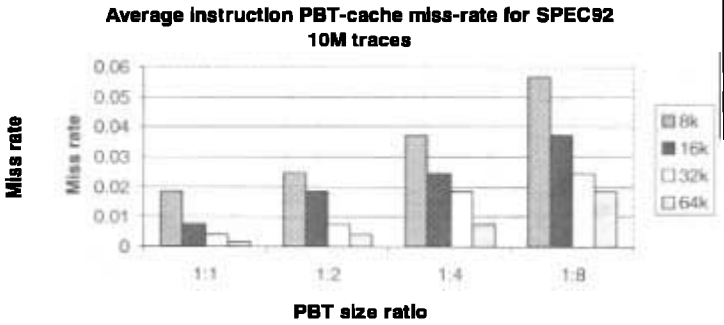

Figure 5: Average miss-rate for direct-mapped instruction PBT-cache, 64 byte blocks. Data averaged from Gcc, Ear, Sc, Swm256, Li, Eqntott, Compress and Tomcatv, all 10M traces.

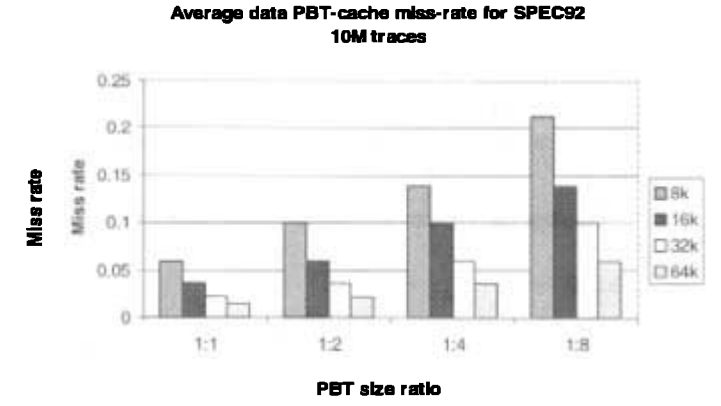

Figure 6: Average miss-rate for direct-mapped data PBT-cache, 64 byte blocks. Data averaged from Gcc, Ear, Sc, Swm256, Li, Eqntott, Compress and Tomcatv, all 10M traces.

2.1.1. PBT Benefit Cost

An important measure is how much we need to improve the clock speed to make a PBT implementation worthwhile, since we are making some cache accesses two cycles. Detailed calculation of benefit cost is in [?]. Table 1 and Table 2 present the PBT benefit cost results.

Cache Size	PBT ratio 1:1	PBT ratio 1:2	PBT ratio 1:4	PBT ratio 1:8
8k	1.77%	2.32%	3.55%	5.42%
16k	0.76%	1.82%	2.40%	3.65%
32k	0.40%	0.77%	1.85%	2.43%
64k	0.16%	0.40%	0.77%	1.85%

Table 1: Minimum instruction cache clock speed-up necessary for benefit of PBT. Assuming lower memory access of eight cycles. Data averaged from Gcc, Ear, Sc, Swm256, Li, Eqntott, Compress and Tomcatv, all 10M traces.

Cache Size	PBT ratio 1:1	PBT ratio 1:2	PBT ratio 1:4	PBT ratio 1:8
8k	5.10%	8.49%	11.79%	18.11%
16k	3.23%	5.32%	8.86%	12.29%
32k	2.03%	3.35%	5.51%	9.18%
64k	1.47%	2.07%	3.41%	5.62%

Table 2: Minimum data cache clock speed-up necessary for benefit of PBT. Assuming lower memory access of eight cycles. Data averaged from Gcc, Ear, Sc, Swm256, Li, Eqntott, Compress and Tomcatv, all 10M traces.

2.2. Cache with PBT and skewing

Next, we add skewing to the cache. Skewing will be implemented using an inter-bank dispersion degree of eight, with a Bit Permute Permutation of identity. The skewing function is shown in Figure 7. A skewing degree of eight has been proven adequate for a 2-way set-associative cache [6]. It is possible to remove the delay of adding skewing by doing the XOR-ing in the address computation cycle. This stage is often less time critical. We can also do the data row selection in the following stage. Skewing will affect the PBT-cache miss-rate indirectly, since the PBT is a subset of the regular cache.

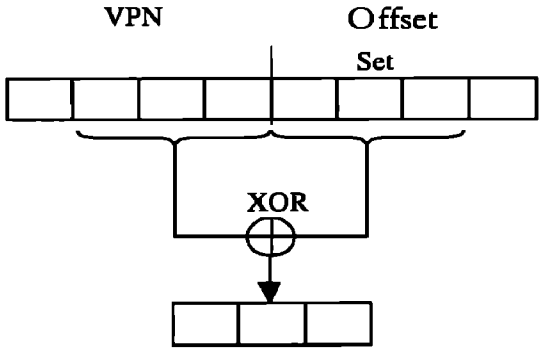

Figure 7: Skewing. Inter-bank dispersion degree of eight. Bit Permute Permutation identity. Simple XOR-function creates the new high order set bits.

2.2.1. Miss-Rate Result from Skewed Cache and PBT

Below is the miss-rate that results from adding skewing to the cache. As we know, skewing attacks the miss-rate by its inter-bank dispersion quality. We are not concerned with the PBT at this time.

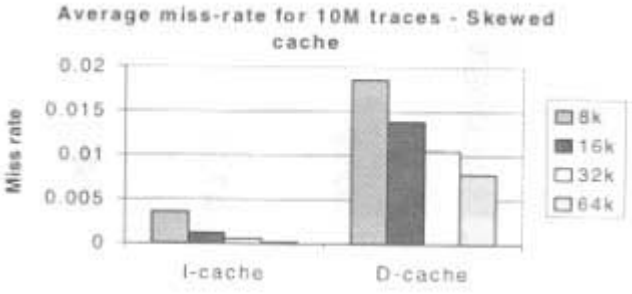

Figure 8: Average miss-rate for 2-way set-associative split cache with skewing degree 8 and 64 byte blocks. Data averaged from Gcc, Ear, Sc, Swm256, Li, Eqntott, Compress and Tomcatv, all 10M traces.

The next simulation results show the PBT instruction/data cache miss-rate, see Figure 9 and 10. Since we achieved a reduction in miss-rate for the instruction/data cache, we should see some reduction in miss-rate for the PBT-cache as well. These reductions are much less though, since we have inclusion between the PBT and the regular cache. The PBT-cache is affected indirectly.

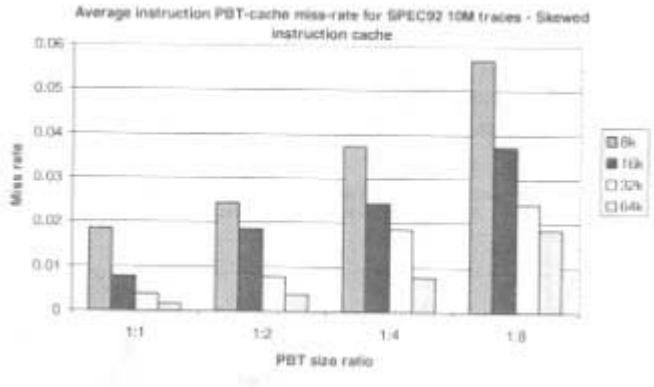

Figure 9: Average instruction miss-ratio for direct-mapped PBT cache with an 8 degree skewed instruction cache and 64 byte blocks. Data averaged from Gcc, Ear, Sc, Swm256, Li, Eqntott, Compress and Tomcatv, all 10M traces.

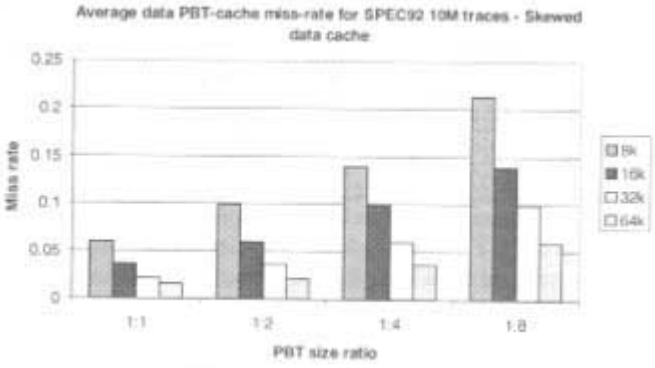

Figure 10: Average data miss-rate for direct-mapped PBT cache with an 8 degree skewed data cache and 64 byte blocks. Data averaged from Gcc, Ear, Sc, Swm256, Li, Eqntott, Compress and Tomcatv, all 10M traces.

From the values in Figure 10, we can see that they are almost identical to the values in Figure 6, where no skewing is used. There is actually a very small difference in the PBT miss-rate, but it is not noticeable from the charts. The biggest saving with using skewing is the reduction in miss rate for the data/instruction cache. Next, we look at the minimum clock speed-up necessary for the use of PBT to achieve some performance improvement.

2.2.2. PBT Benefit Cost with Skewed Cache

From the overall miss-rate improvement caused by the skewing technique, we should expect it to be easier to satisfy the minimum clock speed-up necessary for the PBT technique to be worthwhile. As we can see from Table 3 and 4, the greatest improvement are apparent for the smaller cache sizes, because this is where the skewing technique had the greatest impact.

Cache Size	PBT ratio 1:1	PBT ratio 1:2	PBT ratio 1:4	PBT ratio 1:8
8k	-0.13%	0.42%	1.64%	3.51%
16k	-0.10%	0.98%	1.53%	2.79%
32k	0.27%	0.64%	1.72%	2.30%
64k	-0.16%	0.08%	0.45%	1.53%

Table 3: Minimum instruction cache access speed-up necessary for benefit of PBT. Assuming lower memory access of eight cycles. Data averaged from Gcc, Ear, Sc, Swm256, Li, Eqntott, Compress and Tomcatv, all 10M traces.

We can see from Table 3, that for the largest PBT size we do not need to improve the cycle time at all for the instruction cache. The improved miss-rate due to skewing is so high that we can tolerate two-cycle accesses without any decrease in performance.

Cache Size	PBT ratio 1:1	PBT ratio 1:2	PBT ratio 1:4	PBT ratio 1:8
8k	2.79%	6.18%	9.47%	15.8%
16k	1.68%	3.76%	7.30%	10.7%
32k	1.66%	2.98%	5.15%	8.81%
64k	1.02%	1.53%	2.96%	5.17%

Table 4: Minimum data cache access speed-up necessary for benefit of PBT. Assuming lower memory access is eight cycles. Data averaged from Gcc, Ear, Sc, Swm256, Li, Eqntott, Compress and Tomcatv, all 10M traces.

2.3. Cache with PBT, skewing and indirect-tags

Previously, we have studied and analyzed what happens when we add a cycle time and a miss-rate reducing method. These two methods have both added to the complexity and area cost of the cache. Next we would like to investigate what happens if we add an area cost reducing method. Our main goal here is to answer the question: Can we remove the cost involved with the previously applied methods, without affecting any of the benefits so far? In addition, are there any added benefits through this combination?

An indirect-tagged cache offers more than just reduced area cost. We should also get a faster PBT access time, since the tag comparison time is less, because comparing tag pointers take less time than comparing the whole tag. It is hard to justify how much faster with this assumption since it will be technology dependent. Simulation results assume an LRU-replacement policy, but this will not be possible for a large number of PN-cache entries.

2.3.1. Miss-Rate Results from PBT and Cache with Indirect-tags

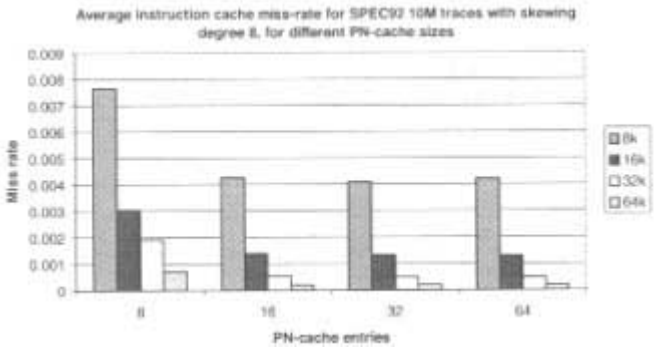

Figure 11: Instruction cache miss-rate with skewing (degree 8) and indirect-tags. Data averaged from Gcc, Ear, Sc, Swm256, Li, Eqntott, Compress and Tomcatv, all 10M traces.

Adding indirect-tags increases miss-rate. This is because there is only a limited set of physical tags possible in the PN-cache. From Figures 11 and 12 we would like to know if the degradation in hit-rate is worse than the improvement accomplished by the skewing effect. For this, we need to compare with the averaged miss-rate for the baseline cache. A PN-cache with 32 entries is sufficient for all cache sizes to keep the miss-rate lower than the baseline cache miss-rate. Beyond 32 entries, there is little increase in performance.

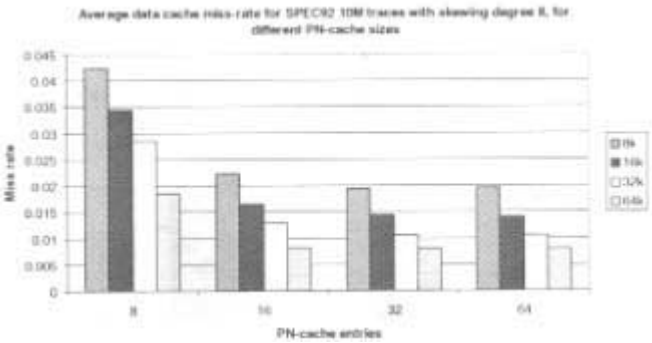

Figure 12: Data cache miss-rate with skewing (degree 8) and indirect-tags. Data averaged from Gcc, Ear, Sc, Swm256, Li, Eqntott, Compress and Tomcatv, all 10M traces.

From Figure 12, we note that for smaller cache sizes, we need more PN-cache entries to keep hit-rate performance levels above the baseline cache. This is due to worse spatial locality for data caches than instruction caches. The thorough reader will notice that for some larger PN-cache sizes, the miss-rate is actually slightly worse. This is because a larger cache requires a larger PN-cache to perform better, since blocks in the cache will stay valid much longer.

The next thing we examine, is how the PBT cache model is affected by the PN-cache. The author believes that a PN-cache with 32 entries is sufficient for most cases. Therefore, we will fix the PN-cache size to this value before conducting any further studies. Figure 13 and 14 shows the miss-rate results for the instruction/data PBT-cache when skewing is used.

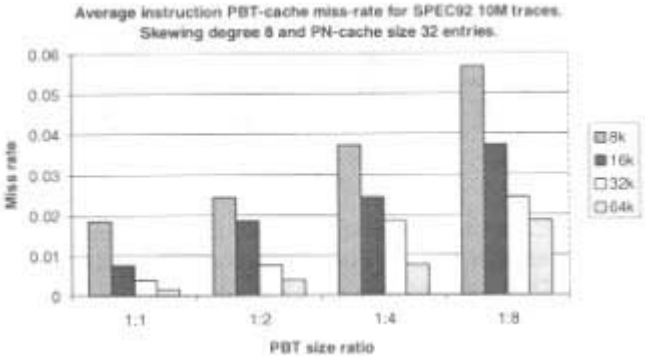

Figure 13: Average instruction cache miss-ratio for direct-mapped PBT cache, 64 byte blocks. Skewing degree 8 and PN-cache size 32 entries. Data averaged from Gcc, Ear, Sc, Swm256, Li, Eqntott, Compress and Tomcatv, all 10M traces.

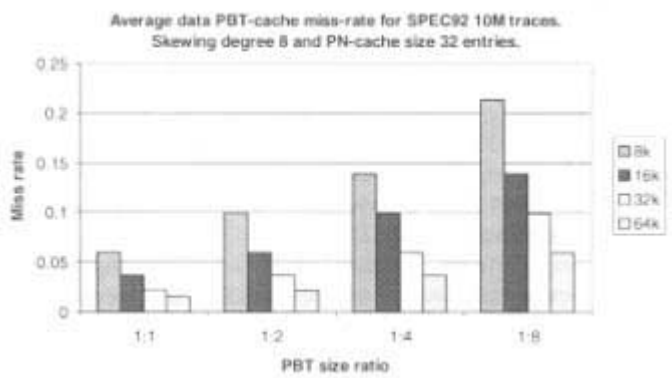

Figure 14: Average data cache miss-rate for direct-mapped PBT cache, 64 byte lines. Skewing degree 8 and PN-cache size 32 entries. Data averaged from Gcc, Ear, Sc, Swm256, Li, Eqntott, Compress and Tomcatv, all 10M traces.

Comparing Figure 13 and 14 with the miss-rate for the cache model without indirect-tags, we see that there are no noticeable differences. A we discussed earlier, a PN-cache with 32 entries will completely fit the workload for the simulated programs.

2.3.2. Area Cost Calculations for Cache with Indirect-tags and PBT

The addition of an indirect-tagged cache offers reduced tag size, in this case for both the cache and the PBT. Next, we try to determine if there is an area cost increase involved with this new cache organization.

Cache Size	PBT ratio 1:1	PBT ratio 1:2	PBT ratio 1:4	PBT ratio 1:8
8k	0.991	0.988	0.986	0.985
16k	0.986	0.983	0.982	0.98
32k	0.985	0.982	0.980	0.979
64k	0.986	0.983	0.981	0.980

Table 5: Areas cost ratio by adding PN-cache (32 entries). Skewing degree 8 and PBT size varying. Address width is 32 bits. Only status bit included is valid bit. Value less than one indicates a reduced area.

Table 5 shows the area cost ratio by adding indirect-tags to the cache. As we can see, there is no increase for any of the cache sizes. The actual benefits are somewhat less than showing due to implementation overhead such as encoders and control logic. The TLB size will also be reduced, since we have removed the physical page numbers and replaced them with an indirect-tag. We have chosen to look at the overall area reduction, not just the reduced tag area, and the reason is that we can better justify adding a PBT and the other techniques. The cost of adding a PBT now basically come at no cost with the added benefit of lower cycle time.

Another important issue is the address width, which is independent of the tag cost. A 64-bit address will only require minimal added cost because of larger tags in the PN-cache, but will give an overall tag area reduction for the cache tag array and the PBT that is overall better.

3. Discussion

For the proposed cache design, there are some drawbacks or disadvantages associated with each of the methods used that we have not yet discussed. This will now be the focus of our attention. From what we have found from this analysis, we also summarize some important findings.

3.1. Effect on miss-rate

We used the skewing technique to effectively reduce the miss-rate in the proposed cache design. The biggest disadvantage of this technique is that we need to use a lot more non-translated address bits to perform the skewing. There are at least a couple of possible solutions to this problem.

(1) Make the page size larger, reducing the cache size or increase associativity.

(2) Allow for more non-translated bits. This works up to a certain point. Seznec [6] showed that up to 18 non-translated address bits is possible without much performance degradation.

(3) As an example, the UltraSPARC microprocessor from Sun Microsystems uses a "next field" prediction scheme to avoid virtual indexing. The high order index bits are predicted. A similar method could be used in this case.

The first step in this design was to add the PBT. Since the PBT does not affect the miss-rate for the cache itself, we were only interested in the added PBT cache miss-rate. As expected for a direct-mapped cache, for small sizes hit-rate performance is poor.

In Table 3 and 4, we found a measure on the difficulty of improving performance with a PBT. The data cache is more difficult to improve because the PBT miss-rate is very poor. A good design alternative in this case is a 32-64Kbyte cache with a PBT ratio between 1:2 and 1:4. We only need to have a PBT cache cycle time that is about 1-5% better than the baseline cache, which is very feasible.

As we have mentioned before, the skewing technique we added improved the miss-rate for the cache but had little effect on the PBT cache miss-rate. The best improvements were made for cache sizes in the range 8-16Kbyte and in particular for the instruction cache. Overall, there was a miss-rate improvement in all cases. This suggests that a smaller cache is better in this case regarding the larger improvements.

In the last step, we added the PN-cache, which effects the miss-rate negatively. Here we also showed that we still gain

from the skewing if the PN-cache is ≥32 entries. In the overall design, miss-rate is improved if PN-cache is greater than 32, for all cache sizes.

3.2. Effect on cycle-time

We used the PBT technique to improve cycle time. For physically tagged caches, we must assume that the TLB access is faster than the PBT. Otherwise, the cycle time will be determined by the TLB, since we need to read the indirect-tags from the TLB to compare with the indirect-tag stored in the PBT. The added cycle-time given by skewing is handled either by doing the skewing function in a previous cycle. On the other hand, if the PBT tag path is longer than the data sub-array access plus the skewing latency, no problem exists, because skewing is not used for the PBT. In the overall design, cycle-time will be improved.

3.3. Effect on area cost

All the methods, except the indirect-tagged cache method, have added to the cache area. A cache consuming larger area can lead to more complex routing and more power consumption, which can indirectly affect cache performance. By implementing a PN-cache, we wanted to overcome or alleviate these problems.

We showed in Table 5 that we can reduce the overall area cost for the complete design. We assumed a 32-bit address width in this case, but the new microprocessors with 64-bits addresses will give us an even greater improvement. Indirectly by reducing area, cycle-time could also be improved.

3.4. The best cache design alternative

Based on the discussion above, the author suggests a cache with the following organization for best performance.

(1) A 32-64Kbyte cache with a PBT size of 1:2 or 1:4.

(2) A minimum of 32 PN-cache entries. In [1] a PN-cache size of 128-512 was simulated. For multi-processor workloads, a much larger PN-cache size is necessary and should be determined based on the cache size as well.

(3) A minimum skewing degree of eight. Simple exclusive-or algorithm sufficient.

(4) Write-Through strategy to avoid problems with page invalidations.

If we can meet the PBT minimum cycle time improvement, this design will have a lower miss-rate, lower cycle-time AND consume less area than the baseline cache. For pipelined processors, it can be difficult to have a cache with multiple-cycle accesses, but the author believes that superscalar processors with a Tomasulo-based scheme will suffer no penalty using this design.

References

[1] Seznec, André. *Don't use page number, but a pointer to it.* Proceeding of the 23rd International Symposium on Computer Architecture.

[2] Hennessy, Patterson. *Computer Architecture, A quantitative approach.* Morgan Kaufmann, 2nd edition, 1997.

[3] Kabakibo, Aiman. Milutinovic, Veljko. Silbey, Alex. Furht, Borko. *A Survey of Cache Memory in Modern Microcomputer and Minicomputer Systems.* IEEE, 1987.

[4] Seznec, André. *Decoupled Sectored Caches: conciliating low tag implementation cost and low miss ratio.* ISCA'21 Proceedings, April 18-21, 1994.

[5] Agarwal, Anant. *Column-Associative Caches: A Technique for Reducing the Miss Rate of Direct-Mapped Caches.* Computer Architecture News, Vol. 21, No. 2, May 1993.

[6] Seznec, André. *A Case for Two-Way Skewed-Associative Caches.* Computer Architecture News, Vol. 21, No. 2, May 1993.

[7] Peir, Jih-Kwon. Hsu, Windsor W. Young, Honesty. Ong, Shauchi. *Improving Cache Performance with Balanced Tag and Data Paths.* Computer Architecture News, Vol. 24, October 1996.

[8] Juan, Toni. Lang, Tomas. Navarro, Juan J. *The Difference-Bit Cache.* Proceedings of the 23rd International Symposium on Computer Architecture.

[9] Bodin, Francôis. Seznec, André. *Skewed Associativity Enhances Performance Predictability.* Computer Architecture News, Vol. 23, No. 2, May 1995.

[10] Wang, Hong. Sun, Tong. Yang, Qing. *CAT – Caching Address Tags, A Technique for Reducing Area Cost of On-chip Caches.* Computer Architecture News, Vol. 23, No. 2, May 1995.

[11] Wulf, Wm. A. McKee, Sally A. *Hitting the Memory Wall: Implications of the Obvious.* Computer Architecture News, Vol. 23, No. 1, March 1995.

[12] Child, Jeff. *Fragmentation ahead for advanced DRAMs.* Computer Design, February 1995.

[13] Wilkes, Maurice V. *The Memory Wall and the CMOS End-Point.* Computer Architecture News, Vol. 23, No. 4, September 1995.

[14] Johnson, Eric E. *Graffiti on "The Memory Wall".* Computer Architecture News, Vol. 23, No. 4, September 1995.

[15] Ohr, Stephan. *Special Report: Cache Design*, Computer Design, 1995.

[16] Jouppi, Norman P. Wilton, Steven J. E. *Tradeoffs in Two-Level On-Chip Caching.* ISCA'21 Proceedings, April 18-21, 1994.

[17] Gee, J. D. Hill, M. D. Pnevmatikatos, D. N. Smith, A. J. *Cache Performance of the SPEC92 Benchmark Suite.* IEEE Micro, page 17-27, August 1993.

[18] Ewy, Benjamin J. Evans, Joseph B. *Secondary Cache Performance in RISC Architectures.* Computer Architecture News, Vol. 21, No. 3, June 1993.

[19] *L2 cache/ controller runs 66-MHz PowerPC.* Page 126, Computer Design, January 1996.

[20] Hill, Mark D. *A Case for Direct-Mapped Caches.* December, 1988.