Brief Announcement: Optimal Bit-Reversal Using Vector Permutations

Anton Lokhmotov, Alan Mycroft
Computer Laboratory, University of Cambridge
15 JJ Thomson Avenue, Cambridge, CB3 0FD
{Anton.Lokhmotov,Alan.Mycroft}@cl.cam.ac.uk

ABSTRACT

We have developed a bit-reversal algorithm (BRAVO) using vector permute operations, which is optimal in the number of permutations, and its cache-optimal version (CO-BRAVO). Our implementation on PowerMac G5 shows 2–4.5 fold improvement for small data sets and 15–75% improvement for large data sets (depending on the data element size) over the best known approach (COBRA).

Categories and Subject Descriptors: C.1.2 [**Processor architectures**]: Multiple Data Stream Architectures (Multiprocessors)—*Single-instruction-stream, multiple-data-stream processors (SIMD)*; F.2.1 [**Numerical algorithms and problems**]: Computation of transforms—*Fast Fourier transform.*

General Terms: Algorithms, performance.

Keywords: FFT, bit-reversal, permutation, SIMD, vector.

1. OVERVIEW

The Fast Fourier Transform (FFT) is an important science and engineering tool. In 1990, Cray Research reported that their 200 machines spent 40% of all CPU cycles computing the FFT [4]. Thus, even modest performance improvements of the FFT are significant in practice.

1.1 Bit-reversal

Many radix-2 FFT algorithms start or end their processing with data shuffled in bit-reversed order. The reordering is typically done by a special subroutine (often called bit-reversal), which can account for 10–30% of the overall FFT computation time [7]. We assume that such a subroutine copies an array $x[\,]$ of $N = 2^n$ elements into an array $y[\,]$ of N elements, such that x and y do not overlap, in the bit-reversed order:

$$y[\sigma_n(i)] = x[i], \text{ for all } i = 0, \ldots, N-1,$$

where $\sigma_n(\cdot)$ reverses bits in a n-bit index. That is, an element of the source array at the index written in binary as $b_0 \ldots b_{n-1}$, is copied to the target array at the index with reversed digits $b_{n-1} \ldots b_0$.

1.2 Vector permutations

A naïve implementation of the N-point bit-reversal performs N loads and N stores (where the order of stores is *not* cache-friendly). Kudriavtsev [6] and Ren [8] reported up to 60% performance improvement for bit-reversal of small (up to 1kB) arrays when using vector permute instructions.

SPAA'07, June 9–11, 2007, San Diego, California, USA.
ACM 978-1-59593-667-7/07/0006.

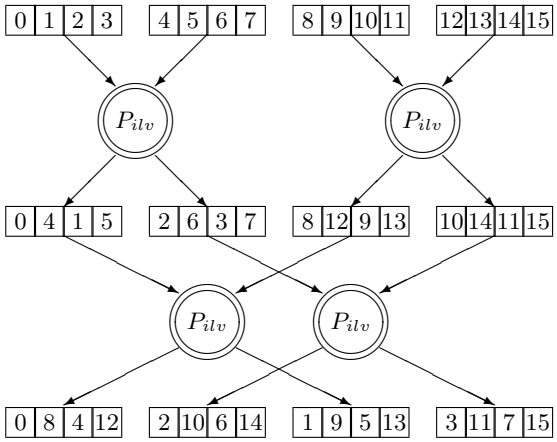


Figure 1: The 16-point bit-reversal using interleaving permutations on 4-element vector registers.

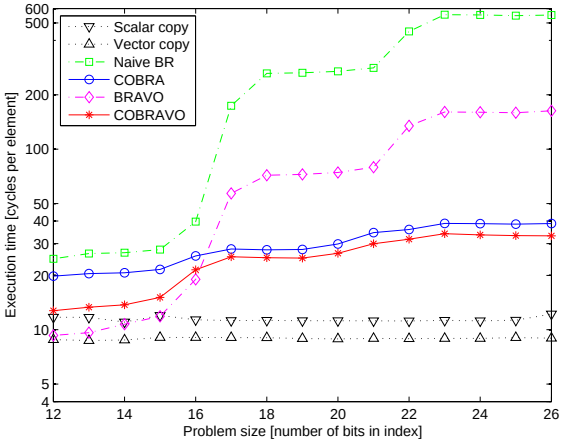
Such instructions are supported by most vector units, including Intel SSE [1] and PowerPC AltiVec [3], as they are valuable for media processing algorithms.

Intrigued by these results, we have analysed the data flow of bit-reversal and derived its structure using vector permute operations, which is optimal both in the number of operations and in the number of distinct control vectors, developing Bit-Reversal Algorithm using Vector permute Operations (BRAVO).

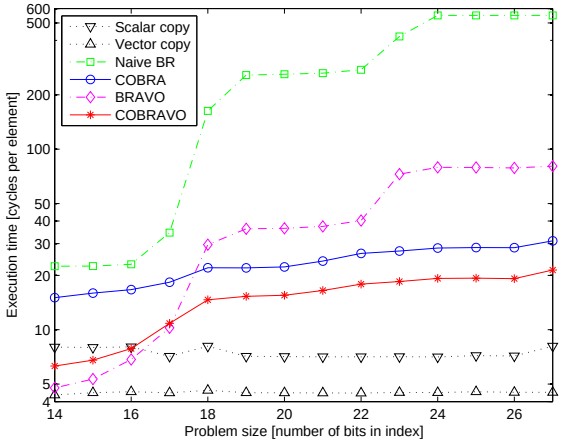
Figure 1 illustrates the optimal 16-point bit-reversal where we assume that the target architecture supports *interleaving* of two 4-element vector registers. (Such interleaving is supported in SSE and AltiVec with `punpckldq`/`punpckhdq` and `vmrglw`/`vmrghw` instruction pairs, respectively.)

To perform the N-point bit-reversal in W-element vector registers, we load N/W source vectors from $x[\,]$, produce N/W target vectors by applying permute operations, and store the target vectors into $y[\,]$. We have proved that the source and target vectors partition into $\pi = N/W^2$ disjoint equivalence classes of W source and W target vectors each: within a class, each source vector provides elements for W target vectors, and each target vector takes elements from W source vectors. (This can be seen for the 16-point bit-reversal, where the only ($\pi = 1$) class encompasses the whole problem.)

Processing each class requires $\log_2 W$ rounds of $\frac{1}{2}W$ interleaving operations each and hence $W(\log_2 W + 1)$ virtual vector registers (which, for example, can be allocated to $W + 1$ physical registers on AltiVec). As the classes are disjoint, we can potentially process them in parallel.



(a) 32-bit data. 16kB buffer (50% of L1 cache)



(b) 16-bit data. 32kB buffer (100% of L1 cache)

Figure 2: Execution time (in cycles per element) against problem size ($n = \log_2 N$)

1.3 Cache-optimal extension

Bit-reversal of large arrays is notorious for its poor cache behaviour [5]. The Cache Optimal Bit-Reverse Algorithm (COBRA) by Carter and Gatlin [2] uses a software buffer to hold in the cache blocks that are otherwise evicted because of associativity conflicts. We consider COBRA the best known approach, since it beat Karp's Hybrid algorithm [5], which had been the best (or near the best) performing method among a collection of 30 bit-reversal algorithms.

Inspired by COBRA, we have developed a cache-optimal extension of BRAVO. We write C, L, and B for the sizes (the number of data elements) of the cache, a cache line, and a buffer block, respectively.

We introduce a software buffer $T[\,]$ to hold a tile of size B^2, where B is selected in a way that $B \geq L$ (so main memory is accessed only once for each cache line) and $B^2 \leq C$ (so most of $T[\,]$ remains in the cache while we permute the tile).

Algorithm 1 Cache-Optimal Bit-Reversal Algorithm using Vector permute Operations (COBRAVO)

Require: $N = 2^n$, $B = 2^b$; $N \geq B^2$, $C \geq B^2$, $B \geq L \geq W$
Ensure: $y[\sigma_n(i)] = x[i]$, for all $i = 0, \ldots, N-1$
1: **procedure** COBRAVO($x[N]$, $y[N]$)
Let $T[B^2]$ be a cache-resident buffer
2: $\tau := N/B^2$ ▷ number of tiles, $\tau \geq 1$
3: $\rho := B/W$ ▷ number of vectors per block, $\rho \geq 1$
4: **for** $t := 0, \tau - 1$ **do** ▷ (parallel) for each tile
5: $t' := \sigma_{|t|}(t)$ ▷ t' is bit-reverse of t
6: **for** $u := 0, B - 1$ **do** ▷ copy sources to buffer
7: **for** $v := 0, B - 1$ **do**
8: $T[u \cdot v] := x[u \cdot t \cdot v]$
9: **for** $l := 0, \rho^2 - 1$ **do** ▷ (parallel) for each class
10: IPBRCLASS(l, $T[B^2]$) ▷ in-place bit-reversal
11: **for** $u := 0, B - 1$ **do** ▷ copy buffer to targets
12: **for** $v := 0, B - 1$ **do**
13: $y[u \cdot t' \cdot v] = T[u \cdot v]$

In lines 6–8 we copy a source tile into the cache-resident buffer $T[\,]$. In lines 9–10 we perform *in-place* bit-reversal of vector classes that form the tile *on the buffer*. Finally, in lines 11–13 we copy the buffer into the target tile.

2. EXPERIMENTAL EVALUATION

In our experiments we used a 2.5 GHz dual-core PowerMac G5 (Model 7,3) with 32kB L1 cache and 512kB L2 cache (both having 128B cache lines).

In Figure 2 we provide evaluation for 32-bit and 16-bit data. A logarithmic scale is used for the Y-axis. As a baseline, we show array to array copy, which is implemented using both scalar and vector memory access. We coded (and partly generated) programs in C (with AltiVec intrinsics) and compiled them using `gcc` 4.0 (at optimisation level O3).

BRAVO outperforms COBRA on small problem sizes by a factor of 2–4.5, and COBRAVO is competitive throughout giving about 15–75% improvement for large data sets (depending on the element size ranging over 32, 16, and 8 bits).

The presented algorithms can be extended with complementary techniques [9] and integrated into FFT algorithms.

3. REFERENCES

[1] A. J. Bik. *The Software Vectorization Handbook. Applying Multimedia Extensions for Maximum Performance*. Intel Press, 2004.

[2] L. Carter and K. S. Gatlin. Towards an optimal bit-reversal permutation program. In *Proc. of FOCS'98*. IEEE.

[3] K. Diefendorff, P. K. Dubey, R. Hochsprung, and H. Scales. AltiVec extension to PowerPC accelerates media processing. *IEEE Micro*, 20(2):85–95, 2000.

[4] J. Johnson and R. Johnson. Challenges of computing the fast Fourier transform. In *Proc. of the Optimized Portable Application Libraries Workshop*, 1997.

[5] A. H. Karp. Bit-reversal on uniprocessors. *SIAM Review*, 38(1):1–26, 1996.

[6] A. Kudriavtsev and P. Kogge. Generation of permutations for SIMD processors. In *Proc. of LCTES'05*. ACM.

[7] C. V. Loan. *Computational frameworks for the fast Fourier transform*. SIAM, 1992.

[8] G. Ren, P. Wu, and D. Padua. Optimizing data permutations for SIMD devices. In *Proc. of PLDI'06*. ACM.

[9] Z. Zhang and X. Zhang. Fast bit-reversals on uniprocessors and shared-memory multiprocessors. *SIAM Journal of Scientific Computing*, 22(6):2113–2134, 2000.

Acknowledgments

The first author gratefully acknowledges the financial support by the TNK-BP Cambridge Kapitza Scholarship and Overseas Research Students Award. We also would like to thank Rok Strniša for providing access to the PowerMac G5 and Manuel Rubio Sánchez for sharing his expertise on bit-reversal.