

Title: Cross verification of SIMD-optimized partial butterfly HM transform
Status: Input Document to JCT-VC
Purpose: Cross check
Author(s): David Flynn (BBC) | davidf@rd.bbc.co.uk
Source: BBC Research & Development

Abstract

This document evaluates the optimization claim of [G497] on SIMD optimizations of the partial butterfly transform[1]. A more accurate method of counting clock cycles was implemented to check software performance. Furthermore, the method was integrated into a larger testbench. The results are found to be accurate and reproducible. When contrasted to the larger benchmark, the G497 solution was found to be a good solution, although not the fastest.

Method

Results shown in tables 1, 2, 3 are based on using a modified version of the supplied software to support more accurate clock cycle measurement. Results are presented on two systems, using intel ICC, targeting SSE4.1:

- Intel Xeon X3450 (2.66GHz Nehalem architecture) CPU
- Intel Corei7 L620 (2.0GHz Nehalem architecture) laptop CPU

When launching the test program on a multi-core system, the test program is bound to an individual CPU to reduce noise from various CPU switching effects.

Clock timings are measured using the CPU's Time Stamp Counter using the RDTSC instruction after issuing a serializing instruction to ensure all outstanding operations have been completed. The cycle cost of reading the TSC and performing all required serialization is measured and subtracted from the results. Using the following inline assembler for GCC:

```
uint32_t __a, __d;
__asm__ volatile(
    "xorl %%eax, %%eax\n"
    "cpuid\n"
    ::: "%eax", "%ebx", "%ecx", "%edx");
__asm__ volatile( "rdtsc\n" : "=a" (__a), "=d" (__d));
__asm__ volatile(
    "xorl %%eax, %%eax\n"
    "cpuid\n"
    ::: "%eax", "%ebx", "%ecx", "%edx");
uint64_t rdtsc = ((uint64_t)__a) | (((uint64_t)__d)<<32);
```

In order to ensure that the CPU is in a stable state. The test program is modified to do sufficient work prior to starting the performance measurements.

Results

Table 1 repeats the results from [2] using an Intel Sandy Bridge series CPU. Results are then normalized against the slowest (32x32) transform to allow comparison.

Tables 2 and 3 show the cross-check results using the supplied software.

Table 1: Results from F447 (Sandy Bridge)

NxN	KxK	Column-row order		Row-column order	
		Estimated cycles	Normalized	Estimated cycles	Normalised
8x8	8x8	167	2.92%	152	2.86%
8x8	4x4	108	1.89%	78	1.47%
16x16	16x16	943	16.49%	859	16.14%
16x16	8x8	547	9.56%	467	8.77%
16x16	4x4	304	5.31%	233	4.38%
32x32	32x32	5720	100.00%	5322	100.00%
32x32	16x16	2764	48.32%	2442	45.89%
32x32	8x8	1732	30.28%	1405	26.40%
32x32	4x4	1167	20.40%	836	15.71%

Table 2: Cross-check X3450 (Nehalem)

NxN	KxK	Column-row order		Row-column order	
		Mean cycles	Normalized	Mean cycles	Normalised
8x8	8x8	132	2.56%	120	2.51%
8x8	4x4	75	1.45%	70	1.46%
16x16	16x16	819	15.87%	749	15.65%
16x16	8x8	467	9.05%	390	8.15%
16x16	4x4	301	5.83%	231	4.83%
32x32	32x32	5159	100.00%	4785	100.00%
32x32	16x16	2576	49.93%	2279	47.63%
32x32	8x8	1609	31.19%	1322	27.63%
32x32	4x4	1181	22.89%	895	18.70%

Table 3: Cross-check L620 Corei7 (Nehalem)

NxN	KxK	Column-row order		Row-column order	
		Mean cycles	Normalized	Mean cycles	Normalised
8x8	8x8	90	1.65%	83	1.99%
8x8	4x4	42	0.77%	37	0.88%
16x16	16x16	694	12.74%	631	15.10%
16x16	8x8	381	7.00%	317	7.59%
16x16	4x4	237	4.35%	178	4.26%
32x32	32x32	5446	100.00%	4178	100.00%
32x32	16x16	2207	40.52%	1989	47.61%
32x32	8x8	1385	25.43%	1118	26.76%
32x32	4x4	999	18.34%	752	18.00%

Additional results

An alternative implementation was written for the 8x8, 16x16 and 32x32 partial butterfly transforms and compiled with GCC-4.6. Table reftab:df-additional shows cycle count measurements for various implementations. To allow fair comparison, the F447 code was integrated into this testbench using both ICC and GCC. All compiled code targets the SSE4.1 instruction set on the 64bit Nehalem architecture.

Timing is calculated by computing the TSC count for each of 10^6 trials, discarding results that are greater than 10% of the mean, then recomputing the mean and standard deviation using these results.

To assist in ensuring a fair comparison, each implementation is inlined to a single function that is aligned to a 4k boundary.

Table 4: Independent implementation X3450 (Nehalem), 10^6 trials

NxN	Transpose		Name	TSC Cycles		
	Input	Output		Min	Mean	Std.Dev.
8x8	.	.	fwd.pb ref_novec	580	584	2.22465
8x8	.	.	fwd.pb ref	414	417	2.0792
8x8	.	.	inv.pb ref_novec	651	657	1.99563
8x8	.	.	inv.pb ref	374	379	2.60802
8x8	.	.	inv.pb opt	111	115	2.496
8x8	.	T	inv.pb opt2	100	104	2.34342
8x8	.	T	inv.pb spiral	114	118	2.81601
8x8	.	T	inv.pb spiral_icc	114	118	2.80017
16x16	.	.	fwd.pb ref_novec	4151	4205	1.96405
16x16	.	.	fwd.pb ref	4163	4216	5.07892
16x16	.	.	inv.pb ref_novec	3848	3862	2.32549
16x16	.	.	inv.pb ref	3845	3872	4.66919
16x16	.	.	inv.pb opt col	843	845	1.76094
16x16	.	.	inv.pb opt3 col	773	775	1.89389
16x16	.	T	inv.pb opt3 col(t)	731	737	2.25768
16x16	T	.	inv.pb opt row	1408	1413	3.12672
16x16	T	.	inv.pb opt splat	1245	1251	2.38126
16x16	.	.	inv.pb opt combined	1054	1070	2.08708
16x16	.	T	inv.pb spiral	737	742	2.33756
16x16	.	T	inv.pb spiral_icc	740	742	2.33424
32x32	.	.	fwd.pb ref_novec	31660	31732	38.6237
32x32	.	.	fwd.pb ref	31493	31532	25.6256
32x32	.	.	inv.pb ref_novec	30176	30196	17.2193
32x32	.	.	inv.pb ref	23945	23971	13.6654
32x32	.	.	inv.pb opt3 col	10920	11136	34.9039
32x32	.	T	inv.pb opt3 col(t)	10297	10560	40.7298
32x32	T	.	inv.pb opt row	5168	5197	22.511
32x32	T	.	inv.pb opt splat	4908	5014	11.1698
32x32	T	.	inv.pb bossen	3894	3900	2.80547
32x32	T	.	inv.pb bossen_icc	3951	3957	3.25911
32x32	.	T	inv.pb spiral	4783	4788	3.91435
32x32	.	T	inv.pb spiral_icc	4771	4775	4.61316

Conclusions

On the basis that the normalised results agree between the three systems, we are happy that the results are valid. The additional results provide a context for comparison, showing that while the F447 results are significantly better than the non-vectorized reference implementation, faster implementations also exist, particularly at 32pt.

References

- [1] A. Fuldseth, L. P. Endresen, S. Selnes, V. Arbatov, M. Püschel, and F. Franchetti, “SIMD-optimized partial butterfly HM transforms.” JCTVC-G497, Nov. 2011.
- [2] A. Fuldseth, L. P. Endresen, S. Selnes, V. Arbatov, M. Püschel, and F. Franchetti, “SIMD optimization of proposed HEVC core transforms.” JCTVC-F447, July 2011.