Intel® Atom™ Processor Performance for DSP Applications

Tim Freeman
N*.A. Software Ltd*

Dave Murray
N*.A. Software Ltd*

Introduction

A recent report [1] compared the performance of several low power processors in DSP applications. This report extends those results to include two processors from Intel's latest family of ultra low power processors, the Atom.

The Atoms are Intel's smallest and lowest power x86 based cpus, and have been designed from the ground up on 45nm technology. The N and Z series Atoms are single core processors which support Intel Hyper-Threading technology, enabling two machine instructions to execute in parallel. Atoms are available with:

- Design power specifications between 0.65 and 2.5W
- Clock speeds of up to 2.0GHz
- Front Side Bus speeds of up to 667MHz
- 512Kb L2 Cache
- SSE3 instruction set

Atom N series (Diamondville generation)

The N270 is currently the most widely used Atom in the netbook market. It features a 1.6GHz clock speed, 533MHz FSB. More recently, Intel have produced the N280, which runs at 1.66GHz with a 667MHz FSB. The N series is compatible with the Intel 945GSE Express chipset (6W) and the 82801GBM I/O Controller (3.3W)

Atom Z Series (Silverthorne generation)

There are 11 cpus in the Z series, which have been designed primarily for ultra-mobile PC's, mobile internet devices, and embedded applications. The higher speed cpus now feature in various netbooks and a number of small form factor boards designed for embedded computing.
The Z series is compatible with the Intel System Controller Hub USWS15L Chipset (2.3W), providing a total TDP of less than 5W.
The report compares times for variety of commonly-used algorithms in high performance DSP applications. All algorithms are implemented using the N.A. Software (NAS) VSIPL (Vector Signal Image Processing Library) API, which has been optimised for Intel SSE architectures. We timed each benchmark code by running 1000 iterations of the library calls; times are in microseconds.

Benchmark Systems

Table 1 outlines the characteristics of the processors we give performance figures for. The 8641D and Penryn figures are repeated from [1] , and we use the same set of algorithms as [1] to give a direct

comparison.

Table 1: Benchmark Test Hardware and Software

Processor	**Freescale MPC8641D**	**Intel Atom (Silverthorne Generation)**	**Intel Atom (Diamondville generation)**	**Intel Core 2 Duo (Penryn Generation)**
Watts (Maximum)	About 25 (no ancillary chipset needed)	About 4.3 inc. ancillary chipset	About 11.8 inc. ancillary chipset	About 31.5 inc. ancillary chipset
Process Technology	90nm	45nm	45nm	45nm
Clock rate	Up to 1.5GHz	Up to 2GHz	Up to 1.67GHz	Up to 2GHz
Cores	2	1	1	2
L1 cache	32KB (each)	32KB	32KB	32KB (each)
L2 cache	1MB (each)	512KB	512KB	Up to 6MB (shared)
Front Side Bus	Up to 600MHz	Up to 533MHz	Up to 667MHz	Up to 1066MHz
Vector Capability	AltiVec (per core)	SSE3	SSE3	SSE4.1 (per core)
Hardware details	Freescale MPC8641D @ 1GHz , 400MHz Front Side Bus (GE Fanuc DSP230)	Intel Atom Z530 @ 1.60GHz, 533MHz Front Side Bus. Intel US15L chipset (Dell Mini 10 laptop)	Intel Atom N270 @ 1.60GHz, 533MHz Front Side Bus Intel GS45E Express mobile chipset	Intel Core 2 Duo SL9400 @ 1.86GHz, 1066MHz Front Side Bus, Intel GS45E Express mobile chipset (HP 2530P laptop)
Software Environment	VXWorks 6.6; GE Fanuc AXISLib VSIPL library rel 2.3.	Linux; N.A. Software VSIPL library for Intel Architecture	Linux; N.A. Software VSIPL library for Intel Architecture	Linux; N.A. Software VSIPL beta for Intel Architecture

1D FFTs

The NAS implementation of VSIPL for Intel Architecture (IA) is multithreaded. For single FFT's we have not shown the threaded timings, though beyond 16K the processor can make use of both cores. However, a single 1D FFT is notoriously difficult to parallelise effectively because of the data movements involved, and multithreading produces only around a 30% performance gain.

For modest lengths, up to 16K, the comparisons reflect primarily the relative processor cycle times. For longer lengths, the major advantage shown by the Intel SL9400 processor reflects the larger L2 cache and higher front side bus speeds on the chips. There is little to choose between the performance of the N270 and Z530, and their relatively small cache affects performance for large FFTs. Nonetheless, they show up very well against the older generation 8641D for large N, reflecting the better memory structure of the Intel family in general..

Single 1D FFT

Table 2a complex to complex 1D in-place FFT: times in microseconds
Times in italic indicate that the data requires a significant portion, or is too large to fit into cache

N	*256*	*1K*	*4K*	*16K*	*64K*	*256K*	*512K*
8641D	2.4	10	71.4	414	2,264	*22,990*	*73,998*
Z530	7.9	32.3	193.6	1,123	*5,562*	*39,982*	*95,058*
N270	7.5	31.7	197.7	1,029	*5,496*	*36,662*	*81,947*
SL9400	1.26	6.3	35.9	197	1,011	4,704	11,732

Figure 2a complex to complex 1D in-place FFT: MFLOPS = 5 N $Log_2(N)$ / (time for one FFT in microseconds)

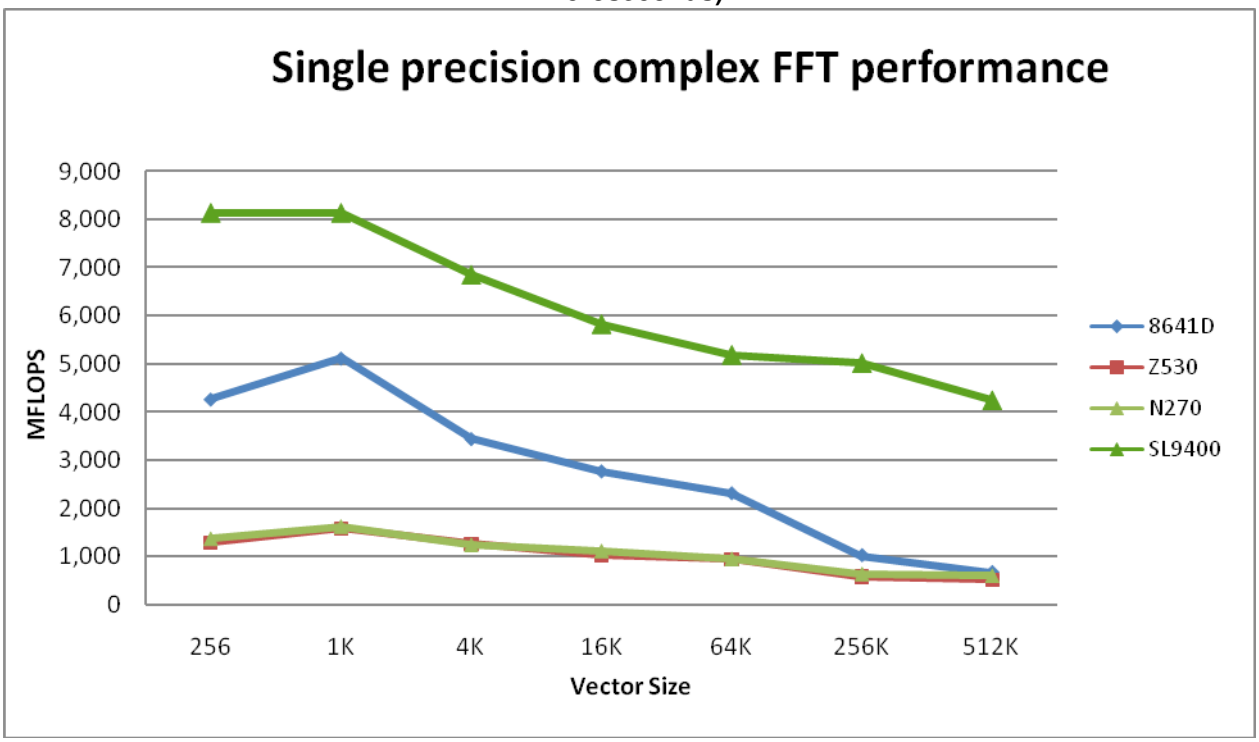


Table 2b real to complex 1D FFT: times in microseconds
Times in italic indicate that the data requires a significant portion, or is too large to fit into cache

N	*256*	*1K*	*4K*	*16K*	*64K*	*256K*	*512K*
8641D	2.1	7.8	46.7	258	1,510	*14,938*	*34,911*
Z530	6.3	27.2	120.8	664	*3,168*	*19,140*	*53,822*
N270	5.3	25.8	116	663	*3,371*	*19,257*	*44,602*
SL9400	1	3.8	21.6	121	535	2,895	5,654

Table 2c complex to real 1D FFT: times in microseconds
Times in italic indicate that the data requires a significant portion, or is too large to fit into cache

N	*256*	*1K*	*4K*	*16K*	*64K*	*256K*	*512K*
8641D	2.1	7.9	45.2	235	1,434	*14,417*	*34,174*
Z530	5.9	27	107.1	637	*3,427*	*19,203*	*53,527*
N270	5.3	25.7	107	645	*3,291*	*18,915*	*44,240*
SL9400	1	4.1	21.5	117	529	2,833	5,702

Multiple 1D FFTs

2D FFTs form a common algorithm in image processing; they reduce to two sets of multiple 1D FFTs separated by a matrix transpose, and it is instructive to look at these two algorithms separately. Table 3 shows the performance obtained for multiple FFTs when contiguous row data is being transformed.

Multiple 1D FFTs multithread well: we have included results for the Intel processors running with 2 threads for a comparison and as expected, the Atom processors show significant gains with smaller ffts, due to the

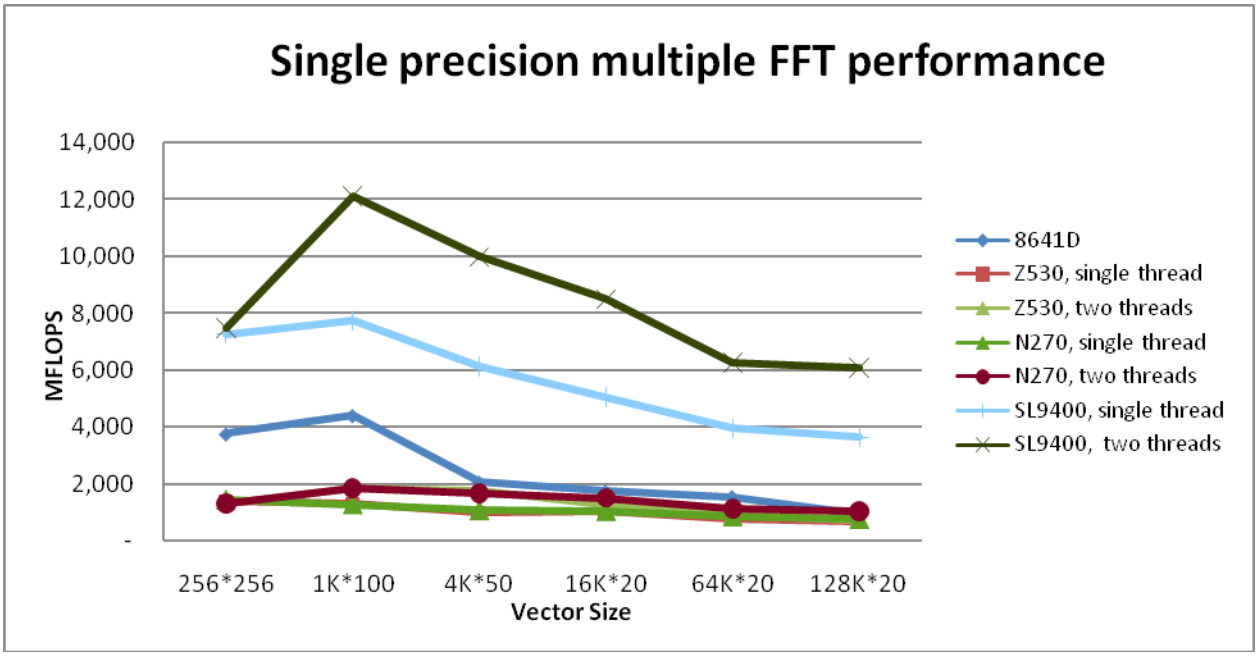
Intel Hyper-Threading Technology built into the single core. Again, for large N the Atom times get surprisingly close to the times for the 8641D.

Table 3 complex to complex multiple 1D in-place FFTs: times in microseconds

Data: M rows of length N; FFT the rows.

*N*M*	*256*256*	*1K*100*	*4K*50*	*16K*20*	*64K*20*	*128K*20*
8641D	698	1,164	5,941	13,111	67,307	231,970
Z530, single thread	1,891	4,010	12,316	22,236	134,020	327,600
Z530, two threads	1,985	2,773	7,006	18,020	114,916	298,004
N270, single thread	1,819	4,003	11,467	21,953	121,257	290,465
N270, two threads	1,976	2,744	7,270	15,146	92,998	214,023
SL9400, single thread	361	661	2,004	4,552	26,577	61,178
SL9400, two threads	350	423	1,232	2,701	16,697	36,653

Figure 3 complex to complex multiple 1D in-place FFTs: MFLOPS = 5 N Log_2(N) / (time for one row FFT in microseconds)
Data: M rows of length N; FFT the rows.



Matrix Transpose

The results for matrix transpose calls for various N and M values are shown in Table 4a and 4B. Matrix transpose is a memory-limited operation on these processors, and the faster front side bus, and larger cache size, gives the Intel processors a major advantage for large matrices. We saw this previously for the SL9400; we now see that the Atom too outperforms the 8641D for large matrices, with the N270 performing particularly well.

Table 4a real matrix transpose: times in microseconds
Data: M rows of length N

*N*M*	*256*256*	*1024*100*	*4K*50*	*16K*20*	*64K*20*
8641D	166	190	2,700	5,591	32,728
Z530	813	1,806	3,567	5,224	20,469
N270	190	1,260	2,644	3,568	14,375
SL9400	81	90	352	478	5,764

Table 4b complex matrix transpose: times in microseconds
Data: M rows of length N

*N*M*	*256*256*	*1024*100*	*4K*50*	*16K*20*	*64K*20*
8641D	877	902	6,744	11,859	65,511
Z530	752	997	8,316	12,129	40,797
N270	589	1,006	5,885	7,422	30,173
SL9400	168	274	705	957	1,608

Vector Arithmetic

We distinguish two classes of vector operations:

1. Simple arithmetic, eg V3 := V1 + V2. For these operations it is possible to achieve near-optimal efficiency on long vectors.
2. Transcendental functions, eg V1 := sin(V2). For these operations, library routines go to a lot of trouble to provide better efficiency, with the use of assembler blocks being common. Quite small details of the processor instruction set can make substantial differences.

For both classes of routine, multithreading is very straightforward, indeed trivial, but again to avoid bias caused by the lack of multithreading in the 8641D library we report timings only on one core in each case.

Simple Vector Arithmetic

Tables 5a and 5b show typical comparisons for "easy" vector arithmetic routines. Note that *italicized* values in the following tables indicate that the data requires a significant proportion of, or is too large to fit into, a processor's L2 cache. Tables 5a and 5b makes it clear just how much difference large L2 caches make. For large N the larger cache available with the Intel SL9400 makes the battle very unequal indeed. This is because the complex vector multiply calculation repeatedly works on the same area of memory—-for N=32K, nearly 1MB is required (32KB * sizeof(complex) * 3 vectors). For N=128K, 3 MB of memory are required. The Atom performances scale well at and below N=32K since a single core has 512Kb of L2 cache. For greater values of N, the data is larger than its cache, resulting in a large number of cache misses and the memory latency penalty of having to access much slower main system memory. Nonetheless, and quite remarkably, in Tables 5a and 5b both Atoms outperform the 8641D for large vectors, while they nearly match the 8641D performance even in Table 5c.
.

Table 5a complex vector multiply v1(i) := v2(i)*v3(i); times in microseconds
Times in italic indicate that the data requires a significant portion, or is too large to fit into cache

N	*256*	*1K*	*4K*	*16K*	*32K*	*64K*	*128K*
8641D	0.78	2.5	18.7	74	145	*3,391*	*9,384*
Z530	1.22	6.14	33.8	144	*542*	*1,236*	*2,236*
N270	1.61	7.84	32.9	134	*328*	*761*	*1,496*
SL9400	0.44	2	8.8	35	75	151	300

Figure 5a complex vector multiply v1(i) := v2(i)*v3(i); MFLOPS = 6 * N / (time for one vector multiply in microseconds)

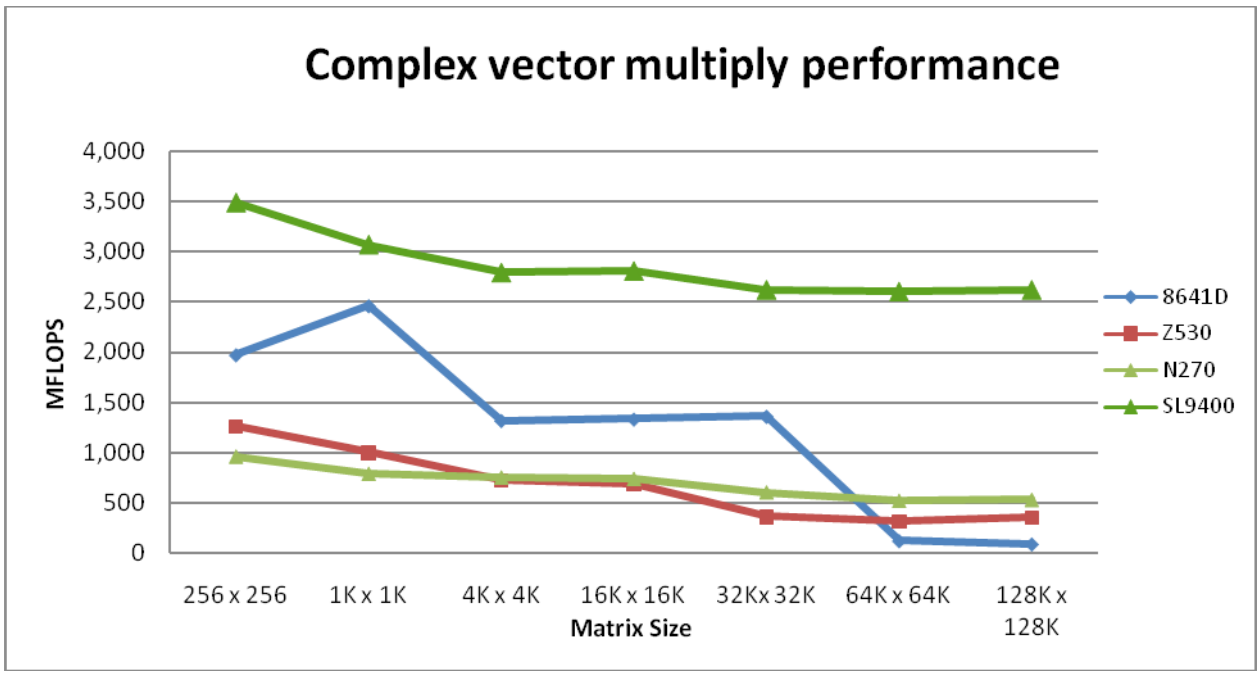


Table 5b real vector multiply/add: v1:= v1+a*v2; times in microseconds
Times in italic indicate that the data requires a significant portion, or is too large to fit into cache

N	*256*	*1K*	*4K*	*16K*	*32K*	*64K*	*128K*
8641D	0.49	1.5	12.5	57.6	110	*1,255*	*7,099*
Z530	0.92	3.07	17.9	74.8	189	*751*	*1,806*
N270	0.91	2.76	16.1	68.4	148	*415*	*1,124*
SL9400	0.21	0.68	3.6	17.6	34	67	134

Table 5c Polar to Cartesian: angle(i) => xy(i); times in microseconds
Times in italic indicate that the data requires a significant portion, or is too large to fit into cache

N	*256*	*1K*	*4K*	*16K*	*32K*	*64K*	*128K*
8641D	3.7	13.8	67.7	268	510	1230	*6255*
Z530	17.5	72.3	257	1,030	2,120	*4,272*	*8,423*
N270	18.7	69	261	1,025	2,056	*4,174*	*8,345*
SL9400	2.7	11.3	44	173	350	694	1,386

Transcendental Functions

Tables 6a-6c show timings for three functions: sin, cosine and square root. As in Ref [1], the PowerPC wins handsomely. This mainly reflects differences in the instruction sets: the PowerPC has built-in select instructions, and integer SIMD support (both used for range reduction) which make implementing efficient trig functions much easier. It also has more registers - this difference becomes very significant when writing assembler blocks to try to speed up these (and other) routines.

The Atoms show up quite badly in these tables, compared with the results for the simpler routines in Tables 5a-c. The difference is large enough that we think it probable we could do better by implementing different algorithms for these routines on Atom: the results here use the same library on all Intel processors.

Table 6a Vector sin: v1(i):= sin(v2(i)); times in microseconds

N	*256*	*1K*	*4K*	*16K*	*32K*	*64K*	*128K*
8641D	0.89	3	11.7	49.5	99	198	468
Z530	6.8	31.3	111	430	859	1,763	3,540
N270	6.7	28.8	114	433	868	1,737	3,493
SL9400	2.3	9.5	37.8	148	293	585	1,174

Table 6b Vector cos: v1(i):=cos(v2(i)); times in microseconds

N	*256*	*1K*	*4K*	*16K*	*32K*	*64K*	*128K*
8641D	0.89	3	11.8	53.6	107	214	497
Z530	6.9	31.3	111	426	856	1,829	3,478
N270	6.8	29.4	108	428	862	1,707	3,448
SL9400	1.7	6.9	28	111	220	447	900

Table 6c Vector square root: v1(i):= sqrt(v2(i)); times in microseconds

N	*256*	*1K*	*4K*	*16K*	*32K*	*64K*	*128K*
8641D	0.52	1.5	6.2	37	74	148	368
Z530	2.52	10.4	46.1	157	316	643	1,304
N270	2.54	10.2	44.2	171	331	676	1,348
SL9400	0.42	1.4	5.8	24	47	92	184

Scatter/Gather

Tables 7a and 7b show representative results for scatter/gather. For small N the timings reflect primarily the relative processor speeds; for large N, again the faster memory access, and the larger cache size on the Intel processors show to advantage, and the Atom performs particularly well against the 8641D.

Table 7a Vector scatter; times in microseconds
Times in italic indicate that the data requires a significant portion, or is too large to fit into cache

N	*256*	*1K*	*4K*	*16K*	*64K*	*128K*
8641D	1.9	7.2	35	136	624	*3,329*
Z530	2.28	7.1	31	159	733	*1,195*
N270	3.15	13.1	54	255	940	*1,864*
SL9400	1.08	3.3	14	57	230	465

Table 7b Vector gather; times in microseconds
Times in italic indicate that the data requires a significant portion, or is too large to fit into cache

N	*256*	*1K*	*4K*	*16K*	*64K*	*128K*
8641D	2.1	8.3	42	168	714	*3,414*
Z530	2	7.7	35	165	747	*1,256*
N270	2.5	9.9	43	185	775	*1,488*
SL9400	0.63	2.7	11	44	181	359

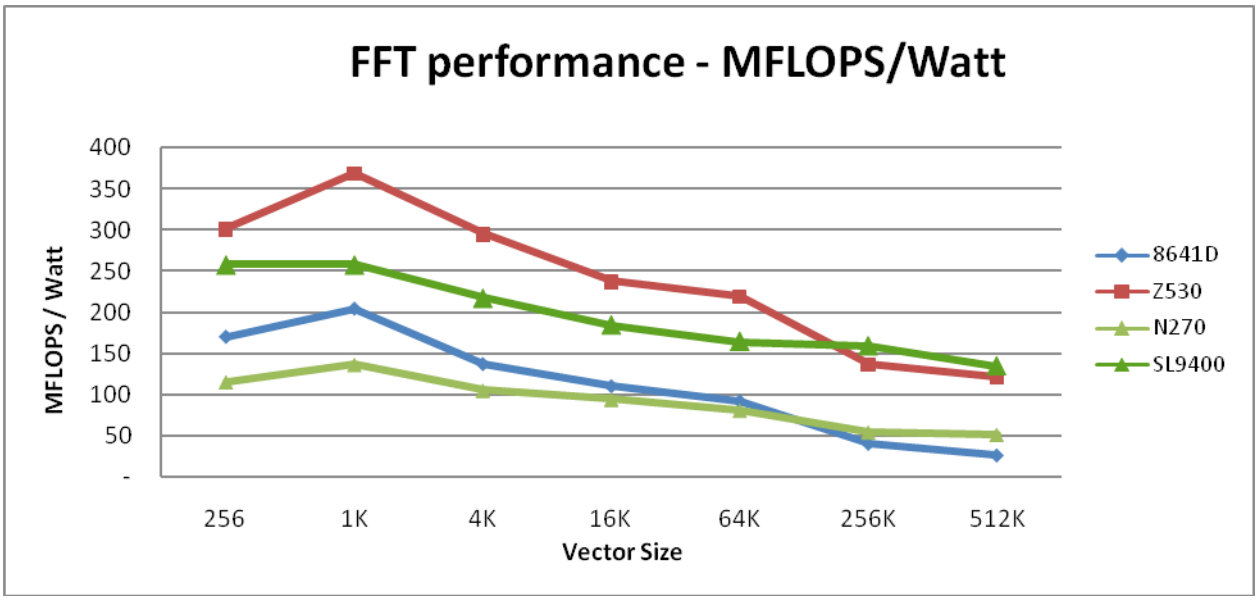
The Future for Atom

Based on these results, the Atom family performs very well in a DSP application context.

Indeed, on a MFLOP/Watt basis the Z530 is a clear winner(see Figure 8 below), largely due to the very small total power consumption of the processor and its associated chipset. Both Atoms perform very well; compared with the higher-power SL9400, the major downside is the handling of large data sets, due to the relatively low L2 cache size, and to a lesser extent the FSB speed. However, as we have seen the Atoms' performance here is still substantially better than that of older generation processors.

And performance/watt is likely to improve rapidly in the near future: The next generation will likely include a system-on-a-chip verson wth integrated graphics. Intel's overall aim appears to be to considerably reduce the overall system power consumption. For low power applications, even the current Atom family performs well on a power/watt basis, as Figure 8 demonstrates. Apart from the obvious advantages of low power consumption in the netbook market, where Atom already dominates, we think that if Intel is aiming Atom at low power, embedded DSP applications it is likely onto a winner.

Figure 8 FFT Performance comparison MFLOPS / Watt



References

1. Using Intel® Processors for DSP Applications: Comparing the Performance of Freescale MPC8641D and Two Intel Core™ 2 Duo processors.

   Mike Delves, N.A. Software Limited and David Tetley, GE Fanuc Intelligent Platforms.

   http://www.nasoftware.co.uk/attachments/018_PPC_Intel_comparison_whitepaper.pdf