NASoftware Limited
Incorporating InfoSAR

ARM VSIPL BENCHMARKS.
Version 1.1

NA. *Software Ltd*
18th December 2019

1. Introduction.

NA Software (NAS) sells low level DSP libraries to board manufacturers and directly to companies within the defence and aerospace industry. The best selling NAS API is the Vector Signal Processing Library (VSIPL) on PowerPC and on a number of Intel platforms including Haswell and Broadwell-DE.

In 2018, NAS also developed a version of our VSIPL DSP library for a range of ARM platforms. This library was further optimised within 2019 and is now a stable reliable software product. The aim of this report is to compare our ARM VSIPL library (running on a ARM A72 system from NXP called the LX2160A) to the performance we obtain on PowerPC and Intel systems.

In carrying out this comparison NAS studied a range of vector and matrix DSP operations as given in the tables below:

Table 1.1: VSIPL Vector operations.

Routine	Description.	Vector lengths
vsip_vadd_f	Float vector addition.	256 to 512K cells in powers of two.
vsip_vmul_f	Float vector multiplication.	256 to 512K cells in powers of two.
vsip_cvmul_f	Complex float vector multiplication.	256 to 512K cells in powers of two.
vsip_ccfftip_f	1D Complex-to-complex in-place FFT.	256 to 512K cells in powers of two.
vsip_ccfftop_f	1D Complex-to-complex out-of-place FFT.	256 to 512K cells in powers of two.
vsip_rcfftop_f	1D Real-to-complex out-of-place FFT.	256 to 512K cells in powers of two.
vsip_crfftop_f	1D Complex-to-real out-of-place FFT.	256 to 512K cells in powers of two.
vsip_vcos_f	Float vector cosine.	256 to 512K cells in powers of two.
vsip_vsin_f	Float vector sine.	256 to 512K cells in powers of two.
vsip_vsqrt_f	Float vector square root.	256 to 512K cells in powers of two.
vsip_vexp10_f	Float vector base 10 exponential.	256 to 512K cells in powers of two.

Table 1.2: VSIPL Matrix operations.

Routine	Description.	2D Matrix Sizes
vsip_madd_f	Float matrix addition.	16^2, 32^2, 64^2, 128^2, 256^2, 512^2, $1K^2$, $2K^2$
vsip_mmul_f	Float matrix multiplication.	16^2, 32^2, 64^2, 128^2, 256^2, 512^2, $1K^2$, $2K^2$
vsip_cmmul_f	Complex float matrix multiplication.	16^2, 32^2, 64^2, 128^2, 256^2, 512^2, $1K^2$, $2K^2$
vsip_ccfft2dip_f	2D Complex-to-complex float in-place FFT.	16^2, 32^2, 64^2, 128^2, 256^2, 512^2, $1K^2$, $2K^2$
vsip_ccfft2dop_f	2D Complex-to-complex float out-of-place FFT.	16^2, 32^2, 64^2, 128^2, 256^2, 512^2, $1K^2$, $2K^2$
vsip_ccfftmip_f	Multiple 1D Complex-to-complex float in-place FFTs.	16^2, 32^2, 64^2, 128^2, 256^2, 512^2, $1K^2$, $2K^2$
vsip_ccfftmop_f	Multiple 1D Complex-to-complex float out-of-place FFTs.	16^2, 32^2, 64^2, 128^2, 256^2, 512^2, $1K^2$, $2K^2$
vsip_mcos_f	Float matrix cosine.	16^2, 32^2, 64^2, 128^2, 256^2, 512^2, $1K^2$, $2K^2$
vsip_msin_f	Float matrix sine.	16^2, 32^2, 64^2, 128^2, 256^2, 512^2, $1K^2$, $2K^2$
vsip_msqrt_f	Float matrix square root.	16^2, 32^2, 64^2, 128^2, 256^2, 512^2, $1K^2$, $2K^2$
vsip_mexp10_f	Float matrix base 10 exponential.	16^2, 32^2, 64^2, 128^2, 256^2, 512^2, $1K^2$, $2K^2$

This report benchmarks the above operations and data sizes with the latest NAS VSIPL DSP libraries on the following platforms:

Table 1.3: Test Platform Information.

Platform 1.	Type:	PowerPC T2080
	Bit:	64
	Operating Frequency:	1.5 GHz
	SIMD Instruction set:	Altivec
	Register width:	128 bit
	Operating System:	Linux
	Physical Cores:	4 (8 with hyper-threading)
	Cores available for benchmarks	8
	Chip Price	$180
	TDP	17W
Platform 2.	Type:	ARM A72 LX2160A
	Bit:	64
	Operating Frequency:	2.0 GHz
	SIMD Instruction set:	ARMv8
	Register width:	128 bit
	Operating System:	Linux
	Physical Cores:	16
	Cores available for benchmarks	16
	Chip Price	$285
	TDP	25W
Platform 3.	Type:	Intel Haswell i7-4700EQ
	Bit:	64
	Operating Frequency:	2.0 GHz
	SIMD Instruction set:	AVX2
	Register width:	256 bit
	Turbo boost:	Disabled
	Operating System:	Linux
	Physical Cores:	4 (8 with hyper-threading)
	Cores available for benchmarks	8
	Chip Price	$378
	TDP	47W
Platform 4.	Type:	Intel Broadwell-DE D-1548
	Bit:	64

	Operating Frequency:	2.0 GHz
	SIMD Instruction set:	AVX2
	Register width:	256 bit
	Turbo boost:	Disabled
	Operating System:	Linux.
	Physical Cores:	8 (16 with hyper-threading)
	Cores available for benchmarks	16
	Chip Price	$555
	TDP	45W

It should be noted that although the T2080 has 8 cores available with hyper-threading, the system only has 4 physical cores. Each pair of hyper-threads on the T2080 shares an Altivec unit on the physical core. This limits the performance gain between using 4 and 8 software cores because the PowerPC maths libraries make use of the significant performance gain offered by using the SIMD Altivec engine.

The benchmark figures presented in this document are average timings presented in microseconds. This is the average time over 100 runs to complete each task and time each function at a particular data size. Therefore, smaller numbers indicate better results.

The programs producing the benchmark timings have all the resources of each platform including hyper-threaded cores available if required. How many cores are used for a benchmark depends on the algorithm being timed and the data size. Algorithms that are serial in nature or have a small data size or computational load will use a single core. Algorithms that are parallelisable and have a large enough data size will use all the cores in the target machine. The timings presented in this report use the optimum number of cores for each function and data size that gives the smallest timing.

This report is organised into the following sections:

Section 1: Introduction – this introduction.

Section 2: VSIPL Vector Performance – compares the performance of the vector DSP operations in the VSIPL library for a wide range of vector lengths.

Section 3: VSIPL Matrix Performance – compares the performance of the matrix DSP operations in the VSIPL library for a wide range of matrix sizes.

Section 4: Conclusions – gives conclusions.

2. VSIPL Vector Performance.

In this section we investigate the performance of the following VSIPL vector functions from a vector length of 256 up to a vector length of 512K in increments of a power of 2:

Vector add;

Vector multiply;

Vector complex multiply;

Vector 1D complex-to-complex in-place FFT;

Vector 1D complex-to-complex out-of-place FFT;

Vector 1D real-to-complex out-of-place FFT;

Vector 1D complex-to-real out-of-place FFT;

Vector cosine;

Vector sine;

Vector square root;

Vector base 10 exponential;

2.1. Float Vector Addition.

In VSIPL the floating point vector add operation is carried out by a call to the library function vsip_vadd_f, after vector views have been created and set. The following table shows the obtained timings in microseconds for vector lengths 256 to 512K. The table contains timings for the PowerPC T2080, ARM A72 (LX2060A) and both the Intel Haswell (i7-4700EQ) and Broadwell (D-1548) systems.

Table 2.1: vsip_vadd_f in microseconds.

System	*256*	*512*	*1K*	*2K*	*4K*	*8K*	*16K*	*32K*	*64K*	*128K*	*256K*	*512K*
T2080	0.45	0.67	1.44	4.08	5.39	15.39	25.2	30.2	35.6	164.33	683.43	1654.76
LX2060	0.15	0.24	0.43	0.88	1.95	9.27	6.76	17.17	15.64	25.63	42.09	51.55
i7-4700	0.07	0.13	0.22	0.39	1.68	3.36	7.21	13.5	22.74	33.53	84.26	232.73
D-1548	0.05	0.1	0.17	0.29	1.23	2.64	5.12	6.15	19.06	27.15	32.37	53.32

The ARM A72 (LX2060A) is significantly quicker than the PowerPC (T2080) for all lengths. The A72 is quicker than the Intel Haswell (i7-4700) at lengths above 8K. Intel Broadwell (D-1548) is quicker than the A72 apart from lengths 64K, 128K and 512K where the A72 is quicker.

2.2. Float Vector Multiply.

In VSIPL the vector multiply operation is carried out by a call to the library function vsip_vmul_f after vector views have been created and set. The following table shows the obtained timings in microseconds for vector lengths 256 to 512K. The table contains timings for the PowerPC T2080, ARM A72 (LX2160A) and both the Intel Haswell (i7-4700EQ) and Broadwell (D-1548) systems.

Table 2.2: vsip_vmul_f in microseconds.

System	*256*	*512*	*1K*	*2K*	*4K*	*8K*	*16K*	*32K*	*64K*	*128K*	*256K*	*512K*
T2080	0.40	0.56	1.47	3.39	4.96	12.61	21.70	27.67	38.74	99.78	669.31	1489.89
LX2160	0.13	0.21	0.36	0.68	1.88	8.81	12.40	16.83	15.61	26.32	43.79	52.64
i7-4700	0.07	0.14	0.23	0.40	1.69	3.37	7.19	15.60	18.54	32.77	58.09	213.88
D-1548	0.05	0.10	0.17	0.30	1.23	2.44	5.00	9.78	13.61	25.87	32.96	52.74

This results is similar to vector add. The T2080 has the slowest performance. The ARM A72 (LX2160A) is quicker than Intel Haswell (i7-4700) for large vector lengths above 32K. However, Intel Broadwell (D-1548) has the best performance. The performance for ARM and Broadwell is the same at a data length of 512K.

2.3. Complex Vector Multiply.

The complex vector multiply operation is carried out with a call to the function vsip_cvmul_f. The following table shows the benchmarks for this function from vector length 256 to 512K.

Table 2.3: vsip_cvmul_f in microseconds.

System	*256*	*512*	*1K*	*2K*	*4K*	*8K*	*16K*	*32K*	*64K*	*128K*	*256K*	*512K*
T2080	0.77	1.41	2.21	6.59	13.71	29.07	39.67	38.49	106.62	661.08	1544.61	3095.3
LX2160	0.33	0.62	1.55	2.57	5.7	11.41	13.58	18.2	28.53	44.11	56.76	114.39
i7-4700	0.12	0.22	0.39	1.98	3.97	8.35	11.64	25.56	59.19	112.17	163.36	1069.55
D-1548	0.08	0.16	0.28	1.42	2.86	5.54	7.73	20.09	39.75	84.93	57.34	145.1

The ARM A72 (LX2160A) is substantially quicker than the T2080 for all vector lengths. The A72 also is quicker than both Intel platforms for large vector lengths above 16K. The A72 performs better against Intel with this function compared vector add/multiply because there is a higher computational load that can be efficiently shared over all the cores in the system. This favours the ARM system because it has more cores than the Intel systems.

2.4. 1D Complex-to-complex In-place FFT.

The table below shows the benchmark figures in microseconds for the function vsip_ccfftip_f. This function performs a 1D complex to complex in-place FFT operation:

Table 2.4: vsip_ccfftip_f in microseconds.

System	*256*	*512*	*1K*	*2K*	*4K*	*8K*	*16K*	*32K*	*64K*	*128K*	*256K*	*512K*
T2080	2.40	4.56	10.00	26.11	50.63	145.85	286.89	388.25	554.78	1257.08	2474.81	14430.40
LX2160	1.54	3.19	7.19	17.39	40.98	110.46	151.28	306.26	418.34	852.13	1144.93	3833.61
i7-4700	0.52	1.11	2.57	6.42	17.69	63.12	198.33	338.03	507.04	1043.75	988.15	2721.33
D-1548	0.38	0.82	2.06	4.70	13.63	40.36	129.52	207.75	292.91	637.21	594.81	1525.68

The ARM A72 (LX2160A) system is quicker than the T2080 for all vector lengths. The Broadwell system (D-1548) has the best performance across all vector lengths. The A72 is quicker than the Intel Haswell (i7-4700EQ) system for vector lengths 16K, 32K, 64K and 128K.

2.5. 1D Complex-to-complex Out-of-place FFT

A 1D complex-to-complex operation is carried out with out-of-place data with a call to the function vsip_ccfftop_f. The following tables shows benchmarks for the function in microseconds:

Table 2.5: vsip_ccfftop_f in microseconds.

System	*256*	*512*	*1K*	*2K*	*4K*	*8K*	*16K*	*32K*	*64K*	*128K*	*256K*	*512K*
T2080	2.37	4.00	10.30	26.56	50.24	139.03	281.30	397.64	745.66	1840.91	3930.37	15231.50
LX2160	1.51	3.34	7.79	18.62	42.89	123.22	137.35	316.89	402.56	982.24	1289.47	3640.39
i7-4700	0.51	1.12	2.46	7.30	20.43	52.00	102.27	259.56	403.56	912.68	1035.05	2878.07
D-1548	0.38	0.81	1.80	5.97	15.86	39.88	84.95	192.84	286.43	688.78	635.31	1544.03

These show both Intel systems performing better than the ARM A72 (LX2160A). However, the A72 has a better performance than the T2080 for all vector lengths. The performance between the A72 and Haswell (i7-4700EQ) is very close at 64K. The 1D FFT algorithms are harder to parallelise efficiently over all the cores in a system than an algorithm such as complex vector multiply. This means that the ARM system does not have such a large advantage with more cores. It is the Intel systems with the larger register width that have the advantage.

2.6. 1D Real-to-complex Out-of-place FFT.

The function vsip_rcfftop_f performs a real-to-complex FFT. The following table shows the timings for vsip_rcfftop_f in microseconds:

Table 2.6: vsip_rcfftop_f in microseconds.

System	256	512	1K	2K	4K	8K	16K	32K	64K	128K	256K	512K
T2080	1.95	3.97	7.2	17.07	42.66	83.26	205.53	410.83	659.83	1831.97	4137.8	9852.75
LX2160	1.26	2.72	5.54	11.85	28.38	65.74	155.15	233.16	462.78	831.52	1481.4	2469.69
i7-4700	0.45	0.83	1.72	3.75	10.31	26.37	65.5	154.27	331.37	554.85	1145.98	1580.87
D-1548	0.32	0.61	1.28	2.78	8.2	18.99	48.91	122.52	237.8	404.69	877.13	1146.36

The Intel systems has a better performance than the ARM A72 (LX2160A). The A72 has a better performance than the T2080. Again the Intel systems have the advantage of the larger register width.

2.7. 1D Complex-to-real Out-of-place FFT.

The following table shows the timings for a 1D complex-to-real FFT (vsip_crfftop_f) in microseconds:

Table 2.7: vsip_crfftop_f in microseconds.

System	256	512	1K	2K	4K	8K	16K	32K	64K	128K	256K	512K
T2080	1.79	3.68	6.35	15.07	38.7	78.4	206.64	418.88	612.89	1419.8	2869.99	8747.67
LX2160	1.29	2.63	5.63	12.12	28.28	64.89	158.04	234.45	452.79	760.02	1543.24	2390.34
i7-4700	0.43	0.86	1.8	3.97	9.22	24.64	64.32	165.98	305.43	523.67	1093.14	1470.77
D-1548	0.32	0.63	1.33	3.08	6.76	18.33	47.75	119.19	222.86	377.88	863.63	1028.39

The Intel systems has a better performance than the ARM A72 (LX2160A) system. The A72 has a much better performance than the T2080 system.

2.8. Float Vector Cosine.

The following table shows the timings for the vector Cosine function (vsip_vcos_f) in microseconds:

Table 2.8: vsip_vcos_f in microseconds.

System	*256*	*512*	*1K*	*2K*	*4K*	*8K*	*16K*	*32K*	*64K*	*128K*	*256K*	*512K*
T2080	2.13	3.97	8.13	15.14	23.09	30.48	47.1	81.12	144.44	274.04	548.79	1395.12
LX2160	2.46	4.87	9.1	14.17	20.63	28.17	37.38	51.14	71.49	110.46	188.07	340.85
i7-4700	0.53	1.51	6.14	4.18	9.69	14.78	21.56	46.86	84.09	152.32	252.62	486.48
D-1548	0.39	0.93	1.42	2.85	9.87	10.51	15.2	20.04	33.46	59.61	65.6	115.74

The ARM A72 (LX2160A) is quicker than the T2080 for vector lengths above 1K and has a similar performance to the T2080 below this length. The T2080 system performs better with trigonometry functions than with other operations that we have looked at in this report. The A72 is quicker than Haswell (i7-4700EQ) for large data lengths above 32K. The Intel Broadwell (D-1548) system has the best performance over all data lengths.

2.9. Float Vector Sine.

The following table shows the timings for the vector Sine function (vsip_vsin_f) in microseconds:

Table 2.9: vsip_vsin_f in microseconds.

System	*256*	*512*	*1K*	*2K*	*4K*	*8K*	*16K*	*32K*	*64K*	*128K*	*256K*	*512K*
T2080	2.08	3.84	7.82	14.64	22.77	30.11	46.54	79.13	140.33	265.99	531.26	1602.55
LX2160	2.31	4.58	9.12	15.29	20.67	25.59	36.94	53.2	69.14	105.82	178.67	324.36
i7-4700	0.53	1.07	2.4	3.07	5.04	14.4	19.59	25.71	45.36	77.82	217.03	274.07
D-1548	0.39	0.8	1.77	2.41	3.81	10.28	13.58	20.25	34.01	59.64	65.13	115.79

These results are similar to the vector cosine results.

2.10. Float Vector Square Root.

The following table shows the timings for the vector Square Root function (vsip_vsqrt_f) in microseconds:

Table 2.10: vsip_vsqrt_f in microseconds.

System	*256*	*512*	*1K*	*2K*	*4K*	*8K*	*16K*	*32K*	*64K*	*128K*	*256K*	*512K*
T2080	0.67	1.12	2.51	4.6	9.37	18.75	25.78	35.7	56.64	73.63	349.76	1199.14
LX2160	0.63	1.11	2.38	4.16	9.31	15.01	21.12	28.15	38.26	51.33	71.21	108.54
i7-4700	0.15	0.27	0.6	1.18	2.35	4.72	9.39	15.84	17.94	29.02	71.41	131.51
D-1548	0.1	0.19	0.41	0.81	1.61	3.23	6.43	6.1	12.76	19.65	26.02	44.33

The ARM A72 (LX2160A) is quicker than the T2080 above 4K and has the same performance below 4K. The A72 is quicker than Haswell (i7-4700EQ) for large data lengths above 128K. The Intel Broadwell (D-1548) system has the best performance over all data lengths.

2.11. Float Vector Base 10 Exponential.

The following table shows the timings for the vector exponential function (vsip_vexp10_f) in microseconds:

Table 2.11: vsip_vexp10_f in microseconds.

System	*256*	*512*	*1K*	*2K*	*4K*	*8K*	*16K*	*32K*	*64K*	*128K*	*256K*	*512K*
T2080	1.04	1.79	3.92	6.48	14.77	19.07	26.32	46.45	76.35	96.2	324.73	1253.5
LX2160	1.85	2.42	8.57	7.38	14.71	24.84	33.53	44.83	62.06	91.72	150.46	265.88
i7-4700	2.74	5.46	13.92	16.34	24.36	42.94	44.43	75.11	138.45	263.73	513.39	1016.01
D-1548	1.81	3.6	11.09	11.52	15.12	20.93	27.84	42.89	71.73	128.71	246.94	478.94

The ARM A72 (LX2160) is quicker than the all the other platforms for vector lengths above 32K. The T2080 has the best performance short vector lengths below the 16K vector length with this function. This is due to the Altivec instruction vec_expte (Vector log raised to the exponent) within the Altivec instruction set.

3. VSIPL Matrix Performance.

In this section we investigate the performance of the following VSIPL matrix functions with data sizes 16x16, 32x32, 64x64, 128x128, 256x256, 512x512, 1Kx1K and 2Kx2K:

Matrix add;

Matrix multiply;

Matrix complex multiply;

Matrix 2D complex-to-complex in-place FFT;

Matrix 2D complex-to-complex out-of-place FFT;

Matrix multiple 1D complex-to-complex in-place FFT;

Matrix multiple 1D complex-to-complex out-of-place FFT;

Matrix cosine;

Matrix sine;

Matrix square root;

Matrix base 10 exponential;

3.1. Float Matrix Addition.

In VSIPL the floating point matrix add operation is carried out by a call to the library function vsip_madd_f, after matrix views have been created and set. The following table shows the obtained timings in microseconds for a wide range of data sizes. The table contains timings for the T2080, ARM A72 (LX2160) and both the Intel Haswell (i7-4700EQ) and Broadwell (D-1548) systems.

Table 3.1: vsip_madd_f in microseconds.

System	*16x16*	*32x32*	*64x64*	*128x128*	*256x256*	*512x512*	*1Kx1K*	*2Kx2K*
T2080	0.53	2.24	5.95	33.34	40.37	714.01	3304.52	13269.6
LX2160A	0.17	0.45	1.88	11.58	15.77	43.51	115.16	1904.85
i7-4700eq	0.09	0.24	1.7	7.17	26.26	63.85	883.01	4483.49
D-1548	0.07	0.18	1.25	5.05	17.62	29.78	139.15	2665.48

The ARM A72 (LX2160A) system is substantially faster than the T2080 for all data sizes. For small data sizes both Intel platforms are quicker than the A72. However, at 256^2, $1K^2$ and $2K^2$ the A72 system is much quicker than both Intel machines.

3.2. Float Matrix Multiply.

In VSIPL the floating point matrix multiply operation is carried out by a call to the library function vsip_mmul_f, after vector views have been created and set. The following table shows the obtained timings in microseconds. The table contains timings for the T2080, ARM A72 (LX2160A) and both the Intel Haswell (i7-4700EQ) and Broadwell (D-1548) systems.

Table 3.2: vsip_mmul_f in microseconds.

System	*16x16*	*32x32*	*64x64*	*128x128*	*256x256*	*512x512*	*1Kx1K*	*2Kx2K*
T2080	0.48	1.89	5.57	31.34	40.11	673.96	2973.35	12036.5
LX2160A	0.15	0.38	1.82	11.45	15.79	43.04	114.91	1627.02
i7-4700eq	0.09	0.24	1.7	7.14	15.78	82.45	932.13	4602.7
D-1548	0.07	0.18	1.25	5.06	13.87	32.56	132.28	2703.23

The ARM A72 (LX2160A) system is faster than the T2080 for all data sizes. For small data sizes the both Intel platforms are quicker than the A72. However, at 1K and 2K the A72 system is quicker than even Broadwell (D-1548).

3.3. Complex Matrix Multiply.

The following table shows the timings for the complex matrix multiply vector function (vsip_cmmul_f) in microseconds:

Table 3.3: vsip_cmmul_f in microseconds.

System	*16x16*	*32x32*	*64x64*	*128x128*	*256x256*	*512x512*	*1Kx1K*	*2Kx2K*
T2080	0.85	2.64	14.59	130.87	272.43	3501.21	13582.9	53015.3
LX2160A	0.33	1.07	5.05	20.26	108.01	404.17	3057.03	12188.6
i7-4700eq	0.15	0.45	4.78	23.78	104.55	471.03	3025.42	12128.2
D-1548	0.1	0.33	3.27	17.38	77.27	310.12	2776.63	11129.6

The above results again show that the ARM A72 (LX2160A) has a much better performance than the T2080. The Intel Broadwell (D-1548) has the best performance over all data sizes. The performance of the A72 and the Haswell (i7-4700EQ) systems is similar above a data size of 32^2.

3.4. 2D Complex-to-complex In-place FFT.

The following table shows the timings for the 2d complex-to-complex in-place FFT function (vsip_ccftf2dip_f) in microseconds:

Table 3.4: vsip_ccfft2dip_f in microseconds.

System	*16x16*	*32x32*	*64x64*	*128x128*	*256x256*	*512x512*	*1Kx1K*	*2Kx2K*
T2080	6.32	18.32	69.21	283.54	679.5	1865.05	16229.3	71812.4
LX2160A	1.8	7.24	31.9	212.31	582.26	1301.06	3426.55	18568.3
i7-4700eq	1.26	4.13	13.98	91.49	395.29	701.3	3529.84	22274
D-1548	0.93	3.06	10.59	61.36	302.42	418.91	1249.38	12031.4

The D-1548 system has the best performance over all data sizes. The T2080 has the slowest performance over all data sizes. The ARM A72 (LX2160A) has a better performance than Haswell (i7-4700EQ) above 512^2.

3.5. 2D Complex-to-complex Out-of-place FFT.

The following table shows the timings for the 2d complex-to-complex out-of-place FFT function (vsip_ccfft2dop_f) in microseconds:

Table 3.5: vsip_ccfft2dop_f in microseconds.

System	*16x16*	*32x32*	*64x64*	*128x128*	*256x256*	*512x512*	*1Kx1K*	*2Kx2K*
T2080	6.56	18.11	68.98	283.2	689.78	2850.32	19898.6	74158.5
LX2160A	1.93	7.64	35.85	213.84	557.46	1277.87	3639.87	17991.2
i7-4700eq	1.43	4.36	14.93	82.98	352.86	586.1	4725.21	23157
D-1548	1.02	3.21	11.34	62.55	281.77	370.54	1399.85	13126.1

The D-1548 system has the best performance over all data sizes. The T2080 has the slowest performance over all data sizes. The ARM A72 (LX2160A) has a better performance than Haswell (i7-4700EQ) above 512^2.

3.6. Multiple Complex-to-complex In-place FFT.

The following table shows the timings for the a multiple complex-to-complex in-place FFT operation (vsip_ccfftmip_f) in microseconds. In this example the algorithm is performing multiple FFTs along the rows of each matrix.

Table 3.6: vsip_ccfftmip_f in microseconds.

System	*16x16*	*32x32*	*64x64*	*128x128*	*256x256*	*512x512*	*1Kx1K*	*2Kx2K*
T2080	4.8	12.35	41.38	129.53	119.94	648.92	4110.62	16808.7
LX2160A	1.07	3.99	16.56	39.05	82.35	176.22	635.84	3075.9
i7-4700eq	0.91	2.9	8.05	46.52	127.28	253.18	981.8	5344.43
D-1548	0.65	2.17	6.07	27.19	55.44	185.07	683.67	3126.19

The T2080 has the slowest performance over all data sizes. The ARM A72 (LX2160A) has a better performance than Haswell (i7-4700EQ) above 64x64. The A72 has a better performance than the D-1548 above 256x256.

3.7. Multiple Complex-to-complex Out-of-place FFT.

The following table shows the timings for the a multiple complex-to-complex out-of-place FFT operation (vsip_ccfftmop_f) in microseconds. In this example the algorithm is performing multiple FFTs along the rows of each matrix.

Table 3.7: vsip_ccfftmop_f in microseconds.

System	*16x16*	*32x32*	*64x64*	*128x128*	*256x256*	*512x512*	*1Kx1K*	*2Kx2K*
T2080	5.04	12.72	41.32	130.54	133.53	1491.32	7574.64	24364.7
LX2160A	1.2	4.34	17.49	39.23	59.51	148.91	729.62	4812.63
i7-4700eq	1.07	3.2	8.77	36.64	57.64	191.5	1762.16	7086.17
D-1548	0.76	2.36	6.68	27.65	34.31	92.98	621.28	4018.14

The T2080 has the slowest performance over all data sizes. The D-1548 has the best performance over all data sizes. The ARM A72 (LX2160A) has a better performance than Haswell (i7-4700EQ) above 256x256.

3.8. Matrix Cosine.

The following table shows the timings for the a matrix cosine operation (vsip_mcos_f) in microseconds:

Table 3.8: vsip_mcos_f in microseconds.

System	*16x16*	*32x32*	*64x64*	*128x128*	*256x256*	*512x512*	*1Kx1K*	*2Kx2K*
T2080	2.19	7.76	23.09	47.1	142.34	519.82	2853.9	11421.3
LX2160A	2.18	5.45	19.01	38.3	72.43	189.1	651.3	2514.52
i7-4700eq	0.56	6.08	9.27	17.24	74.5	213.99	824.51	3496.73
D-1548	0.41	1.78	8.6	15.06	33.03	65.21	215.12	1941.75

The D-1548 system has the best performance over all data sizes. The T2080 has the slowest performance. The ARM A72 (LX2160A) system is better than the Haswell (i7-4700EQ) system for data sizes above 128^2.

3.9. Matrix Sine.

The following table shows the timings for the a matrix cosine operation (vsip_msin_f) in microseconds:

Table 3.9: vsip_msin_f in microseconds.

System	*16x16*	*32x32*	*64x64*	*128x128*	*256x256*	*512x512*	*1Kx1K*	*2Kx2K*
T2080	2.11	7.52	22.93	44.32	142.16	552.72	3275.19	12909.1
LX2160A	1.79	9.76	20.71	37.03	69.56	157.31	593.2	2382.61
i7-4700eq	0.55	2.11	5.1	20.37	44.25	215.1	603.76	3465.82
D-1548	0.4	1.55	4.05	13.43	32.07	63.77	210.45	1973.42

These results are similar to the cosine results above.

3.10. Matrix Square Root.

The following table shows the timings for the a matrix square root operation (vsip_msqrt_f) in microseconds:

Table 3.10: vsip_msqrt_f in microseconds.

System	16x16	32x32	64x64	128x128	256x256	512x512	1Kx1K	2Kx2K
T2080	0.75	2.38	9.36	21.01	53.51	337.60	2496.98	9956.47
LX2160A	0.64	2.37	5.34	20.96	36.26	67.37	182.41	1108.86
i7-4700eq	0.17	0.63	2.37	9.40	16.34	72.75	482.57	3415.05
D-1548	0.11	0.42	1.68	6.44	13.38	26.84	75.86	1932.74

The T2080 system has the slowest performance over all data sizes. The ARM A72 (LX2160A) system is better than the Haswell (i7-4700EQ) system for data sizes above 256^2. It is also better than the D-1548 at 2K by 2K.

3.11. Matrix Base 10 Exponential.

The following table shows the timings for the a matrix base 10 exponential operation (vsip_mexp10_f) in microseconds:

Table 3.11: vsip_mexp10_f in microseconds.

System	*16x16*	*32x32*	*64x64*	*128x128*	*256x256*	*512x512*	*1Kx1K*	*2Kx2K*
T2080	1.09	3.44	14.83	29.71	76.35	360.67	2513.44	10211.10
LX2160A	3.84	10.12	18.87	33.74	62.41	150.97	501.36	1928.35
i7-4700eq	2.49	14.05	21.82	69.79	221.10	508.56	2006.09	8110.46
D-1548	1.83	10.08	15.50	28.16	71.83	244.97	934.81	3756.00

In this example the T2080 system has a better performance than the other systems for small data sizes. The ARM A72 (LX2160A) system is quicker than the T2080 for data sizes above 128^2. The A72 is quicker than Haswell (i7-4700EQ) for data sizes above 16^2. The A72 is quicker than the D-1548 system for data sizes above 128^2.

4. Conclusions.

In this study we have compared running a wide range of optimised VSIPL DSP algorithms on the following systems:

Table 4.1: Test platforms.

System	*Model*	*Frequency*	*Power TDP*	*Chip Cost*	*Register Width*	*Cores*
PowerPC	T2080	*1.5 GHz*	*17W*	$180	128 bit	8 with hyper-threading
ARM	A72: LX2160A	*2.0 GHz*	*25W*	$285	128 bit	16
Intel	Haswell: i7-4700	*2.0 GHz*	*47W*	$378	256 bit	8 with hyper-threading
Intel	Broadwell: D-1548	*2.0 GHz*	*45W*	$555	256 bit	16 with hyper-threading

The Intel systems (Haswell and Broadwell) have the advantage of twice the register width compared to the ARM and PowerPC systems. This means they can implement 8-way float SIMD operations (compared to 4-way on ARM and PowerPC) giving the hardware the potential to be twice as quick. The Intel platforms also have much higher Thermal Design Power (TDP) values which may help to give a higher compute performance. However, this will be a disadvantage in applications where low power consumption is a driving factor. The Intel chips also cost more than ARM and PowerPC.

The ARM system has the advantage of more physical cores than the other systems. However, this is not an advantage for algorithms that are serial in nature or algorithms that process small data sizes.

The following table shows the vector lengths where the ARM A72 LX2160A has a better performance than other platforms for each function:

Table 4.2: ARM vector performance gain.

Function	*T2080*	*Haswell*	*Broadwell D-1548*
vsip_vadd_f	All	> 8K	> 256K
vsip_vmul_f	All	> 32K	> 256K
vsip_cvmul_f	All	> 16K	> 16K
vsip_ccfftip_f	All	> 8K	
vsip_ccfftop_f	All		
vsip_rcfftop_f	All		
vsip_crfftop_f	All		
vsip_vcos_f	> 4K	> 32K	
vsip_vsin_f	> 4K	> 128K	
vsip_vsqrt_f	> 4K	> 128K	
vsip_vexp10_f	> 16K	> 16K	> 32K

The ARM system is much faster than the PowerPC T2080 for all vector maths and FFT functions. For trigonometry functions the ARM system has a similar performance to the T2080 for short vector lengths and a much better performance for larger vector lengths.

The Intel systems are quicker than the ARM system for small vector lengths because they have twice the register width and use more power. This performance advantage disappears for larger vector sizes if the function can be parallised efficiently over cores. The 1D FFT operations given above are more serial in nature than the other vector functions. Therefore, the ARM system fails to beat the Intel platform's performance but it does succeed in narrowing the gap significantly. For functions such as vector add/multiply which are more parallelisable in nature, the ARM system beats the performance of the Intel platforms once the vector length is large enough.

The following table shows the matrix sizes where the ARM A72 LX2160A has a better performance than the other platforms for each test function:

Table 4.3: ARM matrix performance gain.

Function	*T2080*	*Haswell*	*Broadwell D-1548*
vsip_madd_f	All	> 128x128	> 512x512
vsip_mmul_f	All	> 256x256	> 512x512
vsip_cmmul_f	All	> 32x32	
vsip_ccfft2dip_f	All	> 512x512	
vsip_ccfft2dop_f	All	> 512x512	
vsip_ccfftmip_f	All	> 64x64	> 256x256
vsip_ccfftmop_f	All	> 256x256	
vsip_mcos_f	All	> 128x128	
vsip_msin_f	All	> 256x256	
vsip_msqrt_f	All	> 256x256	
vsip_mexp10_f	> 128x128	> 16x16	> 128x128

As was the case with the vector functions the ARM system compares extremely well to the PowerPC T2080. The only function where the T2080 has a better performance is the exponential function with very small data sizes. When we compare the ARM system to Intel we now have more functions where ARM eventually beats the performance of Intel if the data size is big enough. This is because the matrix functions are more parallelisable compared to vector functions such as 1D FFTs.

The ARM system compares more favourably against Haswell than with Broadwell. This is because the Broadwell system has twice the number of cores compared to the Haswell system.

In conclusion, for short vector lengths or small matrices the Intel systems have the advantage of the larger register width compared to the ARM system (256 bit vs 128 bit). These small lengths and data sizes also do not parallelise well over multiple cores. As the data size increases the extra physical cores in the ARM system help reduce the Intel advantage. How much of an advantage this gives the ARM system depends on the algorithm being timed and how efficiently it can be parallelised over cores.