

A COMPARATIVE EVALUATION OF DSPs, MICROPROCESSORS AND THE TRANSPUTER FOR IMAGE PROCESSING

M.B.Sandler, L.Hayat, L.Costa & A.Naqvi

Dept. Electronic & Electrical Engineering,
Strand, London, UK.

ABSTRACT

A range of commercially available processors is investigated to evaluate their potential in image processing systems. This examination covers a selection of the more important instructions in use in image processing algorithms, a selection of the algorithms themselves as a simple benchmark, and includes a survey of multiprocessor systems implemented with some of these processors. It is concluded that Digital Signal Processors (DSPs) are better suited to embedded systems but the Transputer may offer the advantage in a research environment, of general usefulness for supercomputing power.

1. INTRODUCTION

There is now a variety of ways in which to implement Digital Signal Processing (DSP) algorithms; whenever possible, a real-time solution is used. However, we may attempt to classify the spectrum of possible solutions into the following three categories:- special purpose architectures, generalized purpose and general purpose architectures.

The special purpose category comprises systems either constructed from low-level building blocks like multipliers, adders and memory or it might include Application Specific Integrated Circuits (ASICs). Amongst these latter, a most exciting recent development is that of the Cathedral software suite [1] for Silicon Compilation of DSP functions and algorithms. This style of implementation has the greatest potential for high speed operation and offers low silicon area and power consumption.

The second category consists of systems which have been implemented using mass-produced processors, such as general purpose microprocessors and special purpose DSP microprocessors. In these there may be one or more processors together with support devices such as multipliers and convolvers. The advantages of this approach are that development time and hence cost may be significantly reduced, development tools are available in a style known to microprocessor engineers, resulting in an accelerated learning phase, and the systems developed for one application may relatively easily be re-programmed with another algorithm for a different purpose. The class of algorithms is however somewhat restricted by the architecture chosen. The use of assembly language programming is gradually dying away as the C programming language becomes available for all

microprocessors, both conventional (eg Motorola 68000) and non-conventional (eg Transputers, and DSPs like the Texas Instruments TMS320 series).

The final category is used to cover non-real time implementations of algorithms, presumably in a high level programming language on a general purpose computer, which could be personal or mainframe. The advantages to this approach are that the algorithm designer has complete flexibility in the operations that may be implemented and may also be able to employ mixed language programming, for example in an application which requires both numerical and symbolic manipulation (eg FORTRAN and Lisp). However, the system is unlikely to operate in real-time and will often have a restricted I/O bandwidth, meaning that processing must normally occur off-line. Such systems are generally used to experiment with and develop new algorithms.

This paper will deal with the second category as this would appear to offer both high performance, and reasonable ease and cost of acquiring tools and skills in small and medium sized companies as well as larger ones.

Image processing will provide the focus for the applicability of this style of implementation. It is gaining great importance as a sub-class of DSP operations (ISDN, satellite and remote sensing, computer and robot vision) and incorporates those instructions of greatest importance in DSP algorithms.

It also exhibits a high throughput requirement. Consider processing a 512x512 pixel monochrome video image in a single frame period; this requires the computation of some 256000 new items of data in 40 msec, or more than 6 million compound operations per second. Typical compound operations are discussed in section 3.

The processors to be discussed themselves fall into 3 classes. These are the general purpose microprocessors, typified by the Motorola 68020[2]; the generalized purpose Digital Signal Processing microprocessors like the TI 320 series (16 bit integer and floating point versions)[3][4], the Motorola 56001 (24 bit integer)[5], and the NEC μ PD77230 (24 bit integer and floating point)[6]; lastly there is the ubiquitous Immos Transputer, examined in its floating point form, the T800[8]. It had been planned to examine also the AT&T DSP family, but unfortunately, it is not possible to extract exact timing information on these processors and evaluation could only have been performed by using a simulator, which was not available.

Table 1 - Instruction timings						
Processor	TMS320c25	TMS320c30	M56001	NEC	T800	M68020
Instruction cycle (nsec)	100	60	97.5	150	33	60
Instruction	Number of cycles					
ADD	1	1	1	1	1	3
SUB	1	1	1	1	1	3
MULT	1	1	1	1	38	28
DIV			1		39	56
ADDF		1		1	6-9	
SUBF		1		1	6-9	
MULTF		1		1	11-18	
DIVF				1	11-18	
LOAD	1	1	1	1	2	5
STORE	1	1	1	1	2	5

2. COMPARISON OF INSTRUCTION EXECUTION

First, we compare the various processors on the basis of individual instructions, chosen to be typical of image processing algorithms. These are numerical (add, multiply, subtract, divide and floating point version thereof) and data access instructions (storing and loading). The speed in terms of the number of instruction cycles are shown in table 1, together with typical instruction cycle speeds. These figures should only be taken as a rough guide, since they will depend strongly on the type and speed of memory used in an implementation; this might be on-chip or off-chip, and in the case of off-chip memory, may involve one or more wait states. In all cases in the table, the best case has been selected.

It is clear that in the arithmetic instructions, which are micro-coded in the Transputer and the 68020, the DSP chips show a clear advantage. It is interesting to note that integer multiplication on the Transputer is by default executed in the normal ALU: it is obviously good programming policy to use floating point multiplication even for integer quantities, when using the T800.

It is not surprising to see that the DSP chips outperform the others in this selection of instructions, as this is the purpose of their design. It is clear from the table that there is not much to choose between the various DSP chips, until instruction cycle timings are taken into account.

More importantly, the two strongest influences in the choice of DSP device should be ease of programming and interface potential (particularly with respect to multiprocessor systems). The execution speed of the T800's floating point operations is fast, but still significantly slower than floating point versions of DSPs. This can probably be explained by the asynchronous coupling between the CPU and floating point unit (FPU), requiring one to wait for the other to become ready. This would also account for the uncertainty over exact execution speeds. This uncertainty must make it difficult to design real-time systems.

It is worth noting that all DSP chips are designed to have some level of internal parallelism (in the case of the NEC device this is in the form of pipelining). This has the implication that many of the instructions in the table can be exe-

cuted together. For example the TMS320c25 can simultaneously perform a multiply/accumulate operation, together with an index register update and a result scaling. Conventional microprocessors do not have this feature and nor do Transputers, except in that floating point operations and communications may occur in parallel with CPU operations. The pay-off is that execution of algorithms is far more efficient than simply the sum of the execution speeds of the individual instructions.

3. COMPARISON OF ALGORITHM EXECUTION

This section is an attempt to benchmark the various processors. Only in the case of the Hough transform does the on-chip multiplier (where present) come into operation.

Each algorithm is implemented, as far as possible, in the same way on the different processors. Assembly language has been used throughout, except for the Transputers where occam has been used. No attempt has been made in any case to optimize the code by using expert assembly language programming techniques.

Only the TMS320c25 and T414 Transputer were actually available for this comparison, so the figures for the other processors were obtained by coding the algorithms and calculating the operational speeds from manufacturers' data sheets. For the T800 Transputer this procedure was adapted slightly, whereby the algorithms were run on the T414, substituting for the floating point instructions with others which would have the same execution time: the timings were then scaled to account for the different instruction cycle speeds. In the case of the TMS320c30, the code was adapted from TMS320c25 code.

All the timings are for the processing of a single 256x256 pixel image. Thresholding, thinning and the Hough transform all exhibit some degree of data dependency. To take account of this, the same simple test images were 'executed' on all processors (real or modelled).

In the case of the Hough transform it is assumed that 10% of the pixels are white (approximately 2500 pixels) and that there are 90 increments in theta (ie the resolution of straight line identification is 2 degrees).

Table 2 - Algorithm timings for 256x256 image						
Processor	TMS320c25	TMS320c30	M56001	NEC	T800	M68020
Instruction cycle (nsec)	100	60	97.5	150	33	60
Algorithm	Execution speed (secs)					
Threshold	0.085	0.066	0.057	0.097	0.35	0.224
Histogram	0.11	0.041	0.038	0.08	0.35	0.228
Sobel	0.249	0.11	0.236	0.55	1.25	1.227
Laplacian	0.19	0.071	0.153	0.45	0.97	0.688
Thinning	0.38	0.193	0.204	0.9	2.025	0.888
Hough	1.367	0.465		0.725	4.75	6.222

Now the advantage of using DSP processors for this class of algorithm is clear. The importance of the internal parallelism is demonstrated even for those algorithms which do not use any multiplications. An unusual feature is that the 68020 is predicted to outperform the T800 even though its clock runs at approximately half the speed.

Comparing amongst the various DSP processors, it should be noted that the investigators have greater experience in working with Texas devices, though the 56001 is still faster than the TMS320c25 which has a similar clock speed. It is also clear that the newer Texas device has advantages to offer over the older even for implementations which do not use its floating point capacity. The only algorithm for which the c30 is not as efficient as the c25 is thresholding (speed reduced only by about 20% for a 40% reduction in cycle time) and this can be explained by the branching instruction taking 4 cycles in the c30, as opposed to only 1 cycle in the c25.

Note that the NEC device is difficult to program in assembly language, due to its wide instruction word, and this may have influenced the timings.

4. COMPARISON OF MULTIPROCESSOR IMPLEMENTATIONS

In this section, we compare various multiprocessor implementations of the same image processing algorithms (wherever possible). Figures of merit have only been obtained for four architectures, two from references and two from work done in the Computer Vision group at King's College. In all cases,

fortunately, the work has been performed on 256x256 pixel images.

In the table below, the various systems are referred to by a coding scheme. System TMS1, the so-called OSMMA system [8], consists of 8 TMS320c25s operating with a 32MHz cycle time. System TMS2 consists of 8 TMS32010 processors operating with a cycle time of 10MHz [9]. System Trans1 [10] consists of 16 T414 15MHz Transputers and is a prototype Supernode [11], now constructed using T800s. System Trans2 [12] consists of 8 T414 20MHz Transputers in a commercially available development system.

As before the thinning operation timings must be regarded only as an indication, since this algorithm is strongly data dependent. With regard to the timings for the Hough transform, the implementation on the OSMMA system is data independent (ie depends only on image size) whereas the timing for the Transputer network is data dependent, and is given for 4K image points (approximately 7% image occupancy).

It is hard to draw conclusions from such a mix of systems, even though they represent only two processor families. It is clear however that the DSP based systems outperform their Transputer counterparts.

Table 3 - Algorithm timings for Multiprocessors				
System	TMS1	TMS2	Trans1	Trans2
Algorithm	Execution speed (msecs)			
Threshold	18		57	
Histogram	16		40	
Sobel	63	105	203	
Laplacian	42			
Thinning	225	580	230	
Hough	128			650

5. CONCLUSIONS

DSP processors, conventional microprocessors and Transputers have been compared on the basis of their ability to perform image processing tasks, with a view to assessing their potential for implementation in real time systems.

A full selection of image processing algorithms has not been attempted, and the algorithms used are those of use mainly in Computer Vision. In particular only one algorithm, the Hough transform, makes use of multiplication. It is intended to address this potential criticism in future work, and thereby to provide a more complete benchmarking. It is also intended that the range of processors under scrutiny be extended.

Transputer systems are generally going to be the more expensive for similar throughput, but have the advantage that they are more flexible in the sense of re-programmability, and a single system might be used for all processing levels, numeric and symbolic, in a Computer Vision system. The Transputer's versatility is such that the same system can then also be used for other tasks, such as matrix equation solution, computer graphics and other 'supercomputing' tasks.

DSP devices definitely exhibit a clear advantage on a performance per processor basis, and in embedded real time systems, they will prove the more cost effective.

The selection of a particular DSP device should not be based solely on data of the type presented. In particular we have omitted all considerations of cost, board area, power consumption and ease of implementation. This is important because it may well be possible, for example, to use two lower cost processors in place of one more expensive one.

REFERENCES

- [1] P.Pype et al., "A Technology Transfer from Research to Development", Proceedings of the 5th Annual ESPRIT Conference, p3-4, Brussels, Nov. 14-17 1988.
- [2] "MC68020, 32 bit microprocessor, User's Manual" 2nd Edition, Motorola Inc.
- [3] "TMS320C25 User's Guide", Texas Instruments.
- [4] "The Third Generation of the TMS320 Family of Digital Signal Processors - Functional Specification", Preliminary Data, Texas Instruments, 1988.
- [5] "DSP56001 Digital Signal Processor User's Manual", Motorola Inc, 1986.
- [6] "NEC μ PD77230 Digital Signal Processor" Product description, NEC.
- [7] "The Transputer Instruction Set - A Compiler Writer's Guide", INMOS Ltd., 1987.
- [8] A.A.Naqvi, "An Efficient Multiprocessor Architecture for Low Level Computer Vision", PhD Thesis, University of London, 1988.
- [9] K.N.Ngan et al., "Parallel image processing system based on the TMS32010 Digital Signal Processor", IEE Proceedings, Vol 134, Pt E, No. 2, March 1987, p119-124.
- [10] A.C.Sleigh et al., "RSRE Experience Implementing Computer Vision Algorithms on Transputers, DAP and DIPOD Parallel Processors", Parallel Architecture and Computer Vision, ed. by I.Page, Clarendon Press, Oxford, 1988.
- [11] C.Jessope et al., "Base Level Software for the P1085 Supernode", Proceedings of the 5th Annual ESPRIT Conference, p796-813, Brussels, Nov. 14-17 1988.
- [12] M.B.Sandler & S.Eghtesadi, "Transputer Based Implementations of the Hough Transform For Computer Vision", Microprocessing and Microprogramming 24 (1988) p403-408, North Holland Press.