

DSP on the desktop

Confronted with selecting a platform for a dsp algorithm, few engineers would consider the modern pc. Yet a pc's cpu clock rate can be much higher than that of most dsps and the vector operations which many pc cpus support are comparable to those implemented in dsps.

A key advantage of using a pc platform for digital signal processing is that most engineers have one available. This means algorithm and software design can proceed with a close approximation to the final hardware – long before it is available. Benchmark implementations of key algorithms quickly show whether or not the platform has the required power.

Another advantage is the performance of pc compilers; whilst dsp compilers are

Signal processing on a desktop pc. By Peter Massam.

Cache systems are, nevertheless, complex and this can be a disadvantage if data processing loops need to be optimised beyond the compiler's capabilities.

DSP algorithms usually require access to a few simple hardware devices, and their drivers are rarely complicated. However, the algorithms are just one part of a system that must usually communicate through more complex channels. This task is frequently left to a separate controller, but if the two elements of the system are to be

engineers who have worked with dsps. Whilst there is a strong similarity between the AMD and Intel extensions, they are not always compatible. It is advisable to examine each set and to choose a processor that best matches the algorithm's requirements.

Beware of subtleties when using these extensions. The most significant is that data for vector operations are loaded from and stored to memory in large chunks (for example, 64bit or 128bit in

frequently very good at optimising loop based code, they are rarely efficient for control code. However, caution is necessary because of the variety of pc processors – support for vector operations may vary greatly and details of the cache system will impact performance.

PCs usually have more memory than a typical dsp based system, which can be an advantage for some dsp algorithms: a modern pc's multiple cache layers means relatively slow (and cheap) memory may be used and the cache system is well understood by compilers, which means they can use it efficiently most of the time.

combined on to a single platform, then a pc based solution becomes attractive.

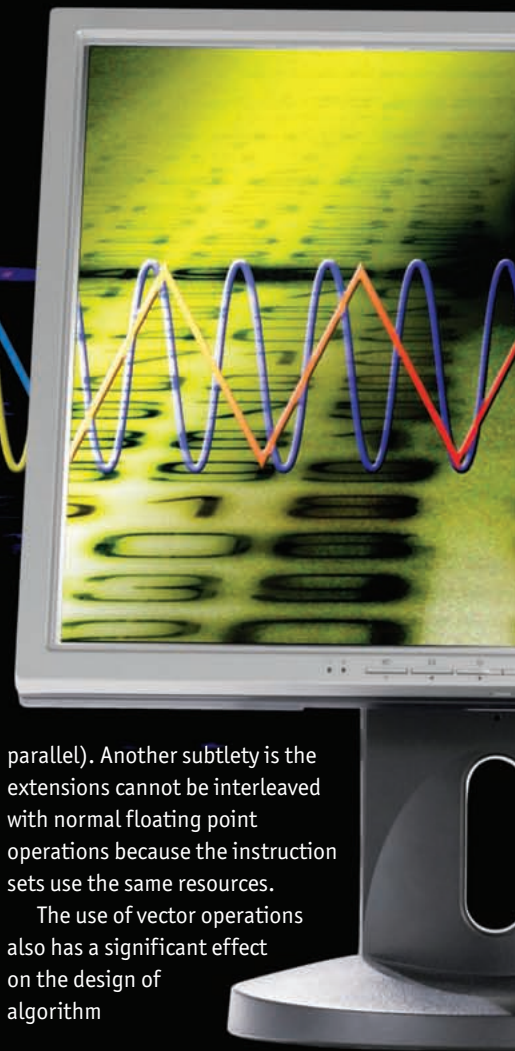
PC vector operations

The growth of gaming has introduced vector operations into pc processor environment, supporting manipulation of the graphics data necessary to present increasingly life like game environments. But this can be used for other dsp algorithms.

There are numerous varieties of vector operations – from Intel and AMD, for example. Vector operations – also known as SIMD, or single instruction, multiple data operations – are familiar to most

parallel). Another subtlety is the extensions cannot be interleaved with normal floating point operations because the instruction sets use the same resources.

The use of vector operations also has a significant effect on the design of algorithm





implementations. The process of adjusting an implementation for a SIMD architecture is known as vectorising.

Memory management is the most significant factor affecting a pc cpu's performance because the basic memory is invariably slow (compared to a dsp system). This affects dsp algorithms and all software running on a pc. Consequently, a deal of ingenuity has gone into designing memory caches to hide slow memory.

The cache system available on different pc cpus varies enormously, but consider the Pentium 4's Level 1 data cache as an example. It is an 8kbyte 'four way set associative' cache with 64byte lines. Essentially, this means that memory is cached in chunks (lines) of 64byte.

aligned to the start of a line, then there will be cache miss for the first byte, but the next 63byte will be in the cache when they are required.

- If data is required in a non sequential order, consideration should be given to the arrangement of the data in memory. In the worst case, cache misses can occur every four accesses if the data is being read from at address intervals of 2048.

There are numerous techniques for optimising cache usage, but PREFETCH is worth mentioning. This forces the cache controller to read the contents of an address into the cache. In a loop, it increases the processing load of the loop, but it may reduce the overall execution time by removing the stalls that follow a cache miss.

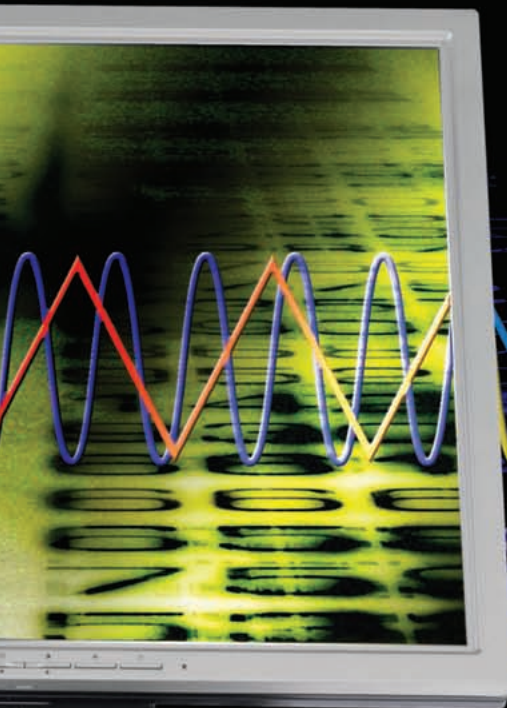
Execution speed is affected by data cache and instruction cache sizes. Arranging the inner loop of an algorithm to fit in a single cache line produces the most efficient code. Whilst most x86 cpus have separate Level 1 data and instruction caches, newer cpu architectures move the instruction cache

2Mbyte Level 3 cache shared between its four cores, each of which has its own 512byte Level 2 cache.

For algorithms requiring high levels of processing power, multicore pc processors are an option. The programming challenge here revolves around how to distribute the algorithm(s) between cores to achieve best performance. If the cores' activity is to be tightly coupled, exchange of data between them becomes critical. This is why we are seeing the appearance of Level 3 caches shared between cores.

If the cores' activity is not tightly coupled, then it becomes more difficult to balance the loading. A common approach is to assign one core to management tasks and the other(s) to dsp algorithms. Whilst this can be inefficient, it is simplest to design and is an example of common engineering compromise between development time and run time efficiency.

There are many compilers from which to choose, but is worth looking for ones supporting the vector operations of the



There are a number of consequences of this architecture:

- Whenever a cache miss occurs, 64byte must be loaded from memory – even if just one byte is required. This carries a processing time cost.

- If a set of data used in an algorithm is

inside the cpu, frequently placing it after the decode unit.

Currently, Level 1 data caches tend to range from 8kbyte to 64kbyte. AMD's cpus tend to have larger caches and fewer 'ways' (two is common), whereas Intel's cpus tend to have smaller caches and more 'ways' (four is common).

Level 2 caches are invariably shared and there is no distinction between data and instructions. Like Level 1 caches, they tend to have 64byte lines, but they usually have more 'ways'. They can store up to several Mbyte of data.

Level 3 caches, however, are quite rare. The AMD Phenom range has a

target processor with 'intrinsics'.

Profiling tools are important when writing dsp software for a pc cpu, particularly ones that can monitor cache behaviour. A good example for Linux platforms is the Valgrind tool suite, which includes the Cachegrind tool

(<http://valgrind.org/>). There are also many libraries that can simplify the development of a dsp implementation and a good example is FFTW libraries (www.fftw.org/). ■

Author profile:

Peter Massam is a senior technical consultant with Plextek.