Application Report
SPRA472

TMS320C6211 Cache Analysis

Abstract

The TMS320C6211's caches deliver high performance without the cost of large arrays of on-chip memory. The efficiency of the TMS320C6211 caches makes low cost, high-density external memory, such as SDRAM, as effective as on-chip memory.

Contents

Figures

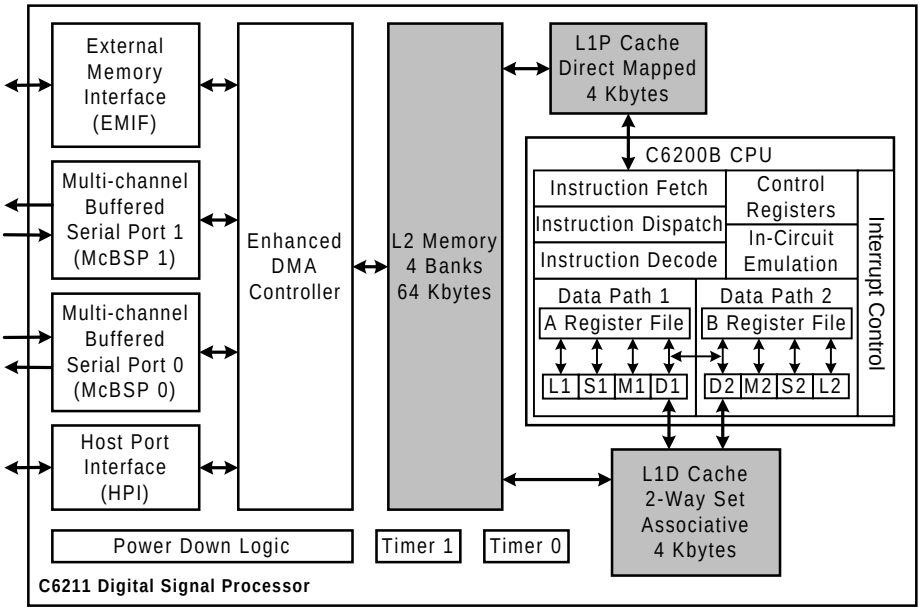
Tables

1 Cache Architecture Overview

The TMS320C6211 ('C6211) utilizes a highly efficient two-level real-time cache for internal program and data storage (See Figure 1). The 'C6211's caches deliver high performance without the cost of large arrays of on-chip memory. The efficiency of the TMS320C6211 caches makes low cost, high-density external memory, such as SDRAM, as effective as on-chip memory. The 'C6211 executes over 99.5% of all CPU cycles without going off-chip. This leads to greater than 80% of the cycle count performance of a C62x device with infinite internal memory.

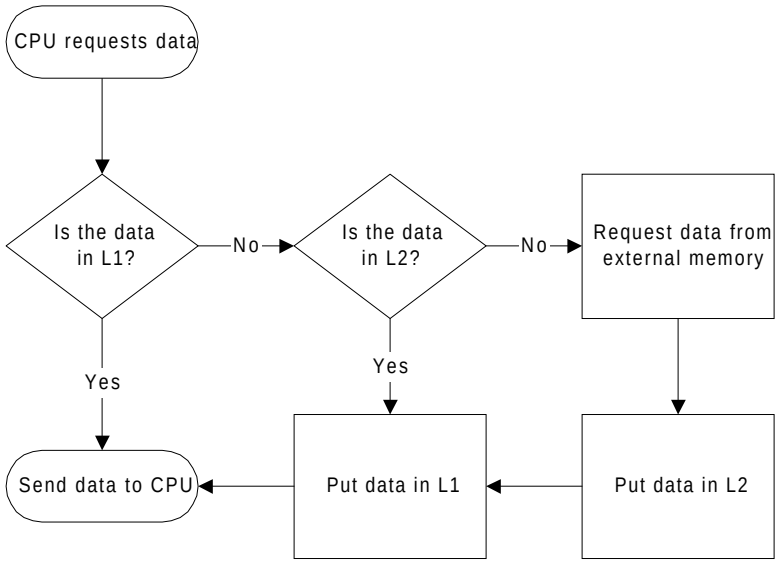
The TMS320C6211 employs a two-level memory architecture for on chip program and data accesses. The first level has dedicated 4 Kbyte program and data caches, L1P and L1D respectively. The second level memory is a 64 Kbyte memory-block that is shared by both the program and data, designated L2. Dedicated L1 caches eliminate conflicts for the memory resources between the program and data busses. A unified L2 memory provides flexible memory allocation between program and data for accesses that do not reside in L1. Since data is frequently resident in L1, flexibility is more important at the L2 level than conflict reduction.

Figure 1. TMS320C6211 Block Diagram



Each cache consists of cache memory, a smaller block of memory to save the state of the cache known as a tag RAM, and a cache controller. When an access is initiated by the CPU, the cache controller checks its tag RAM to determine if that data resides in the cache. If that data does reside in the cache a cache hit occurs and that data is sent to the CPU. If the CPU data does not reside in the cache, then a cache miss occurs. On a cache miss, the controller requests the data from the next level memory. In the case of an L1P or L1D miss, the next level memory is the L2. In the case of an L2 miss, the next level memory is the external memory. The amount of data that a cache requests on a miss is referred to as the cache's line size. Figure 2 illustrates the decision process used by the 'C6211 memory system to fetch the correct data on a CPU request.

Figure 2. TMS320C6211 Two-level cache fetch flow



By using a cache, which dynamically allocates memory to reduce the latency to slower memory, a processor's performance is dramatically increased. A cache's performance can be affected by a situation known as thrashing. For thrashing to occur, data must be read into the cache. Subsequently, another location is cached whose data overwrites the first data. When the first data is requested again, the cache must again fetch it from the slow memory.

1.1 Level-one Program Cache (L1P)

The L1P is organized as a direct mapped cache with a 64-byte line size. A direct mapped cache is well suited for DSP algorithms, which tend to consist of small, tight loops that rarely thrash. The L1P line size provides a modest prefetch of the next fetch packet, eliminating the startup latency for fetching that packet.

In direct mapped cache, every cacheable memory location maps to only one location in the cache. Thus, the cache controller needs to check only one location in the tag RAM to determine if requested data is available in the cache. DSP algorithms primarily consist of loops that execute the same program kernel many times on multiple data locations. Such algorithms remain in a loop for a long time before proceeding to the next kernel. The L1P is large enough to hold several typical DSP kernels simultaneously. Since these kernels execute sequentially, they will not thrash in the L1P. Thus, a simple direct mapped cache is all that is needed to achieve considerable program performance without requiring complex caching hardware.

When a cache miss occurs, the L1P requests an entire line of data from the L2. In other words, both the requested fetch packet and the next fetch packet in memory are loaded into the cache. Since most applications execute sequential instructions, there is a high likelihood that the next fetch packet will be immediately available when it is requested by the CPU. Thus, the startup latency to fetch the next fetch packet is eliminated by bursting an entire cache line. Fetching ahead also reduces the number of cache misses. Eliminating startup latency and reducing misses reduces the execution time of an application considerably compared to a cache with a smaller line size.

1.2 Level-one Data Cache (L1D)

The L1D is organized as a 2-way set associative cache with a 32-byte line size. A set associative cache provides additional flexibility to a direct mapped cache. This cache architecture is beneficial for DSP data, which tends to be more random and have larger strides than program data.

A 2-way set associative cache comprises two direct mapped caches, each of which is referred to as a cache way. Each way in the L1D caches 2 Kbytes of data. In a 2-way set associative cache, every cacheable memory location can reside in one location in each cache way. A 2-way set associative cache reduces the chance of a cache thrash since two thrashing addresses can be stored in the cache simultaneously. This is a beneficial architecture for DSP data, which often accesses multiple arrays simultaneously, such as arrays of coefficients and samples. A 2-way set associative cache is a advantageous design for the TMS320C6211 since the CPU has two data paths, which could simultaneously access two different data arrays. The L1D minimizes the chance of these two data paths thrashing.

The L1D replaces data with a Least Recently Used (LRU) replacement strategy. LRU replacement chooses which set to update with new data by determining which of the two cache ways was accessed least recently. The new data is then placed in the appropriate set of that least recently used way. LRU is the best replacement strategy for set associative caches because of the temporal locality of data – once data has been used it will probably be needed again within a short time. Thus, a cache should always keep data that was most recently used, and replace the least recently used data.

Like the L1P, the L1D line size provides a prefetch of subsequent data, minimizing the fetch latency for that data. When a cache miss occurs, the L1D requests an entire line of data from the L2. When the CPU is fetching a data array from contiguous non-cache memory, this greatly reduces the latency for subsequent data fetches. In the case of an array of words, only the first fetch will experience the delay in going to the L2 or off chip. The next seven array elements will be fetched from the L1D, each in a single cycle. For half-word and byte arrays, the benefit will be even greater since the next 15 or 31 array elements will be in the cache.

The L1D memories are dual ported, which allows the L1D to support two simultaneous CPU data accesses without stalling.

1.3 Level-two Cache/Unified Memory (L2)

The L2 is a 64-Kbyte SRAM divided into four 16-Kbyte blocks. The L2 is a unified memory, used for both program and data. The amount of program or data in the L2 is configurable. For example, if your application requires only 7 Kbytes of program space and 57 Kbytes of data space then both could be linked into the L2 at the same time. Likewise, if your application needed more program space than data, the majority of the L2 RAM could be linked as program space.

Each of the four blocks can be independently configured as either cache or memory mapped RAM. This allows you to dictate the amount of the L2 that is used as cache and how much is used as RAM. If your application uses some data which must be accessed quickly that data can be linked into an L2 block which is configured as RAM. The rest of the L2 can be configured as cache, which will provide high performance operation of the remaining program and data.

When an L2 block is configured as RAM, external data is not cached in that block; instead, that memory is accessed by direct addressing. Each block that is configured as a cache adds a cache way to the L2. For example, when only one block is configured as cache, the L2 operates as a 1-way set associative (direct mapped) cache and 48 Kbytes of RAM. When all four blocks are configured as cache, the L2 operates as a 4-way set associative cache. Figure 3 and Figure 4 illustrate the division of the L2 memory space according to the L2 Mode.

Figure 3. L2 Memory Configurations, Diagram 1

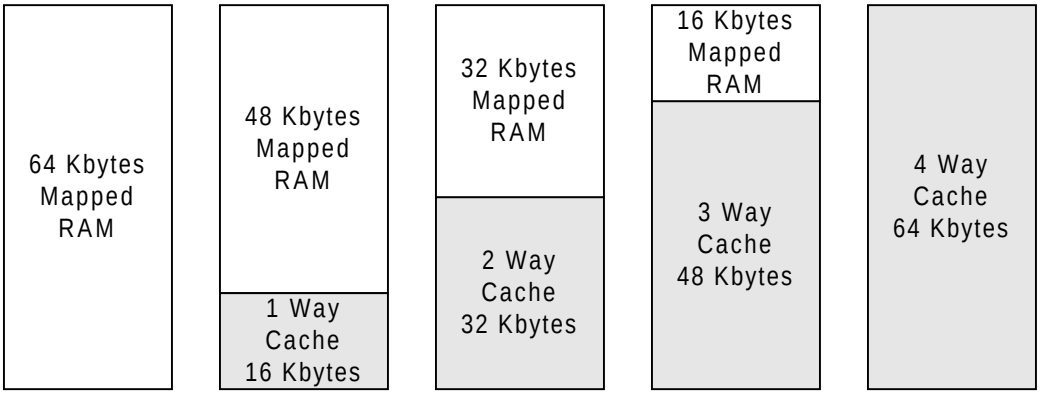
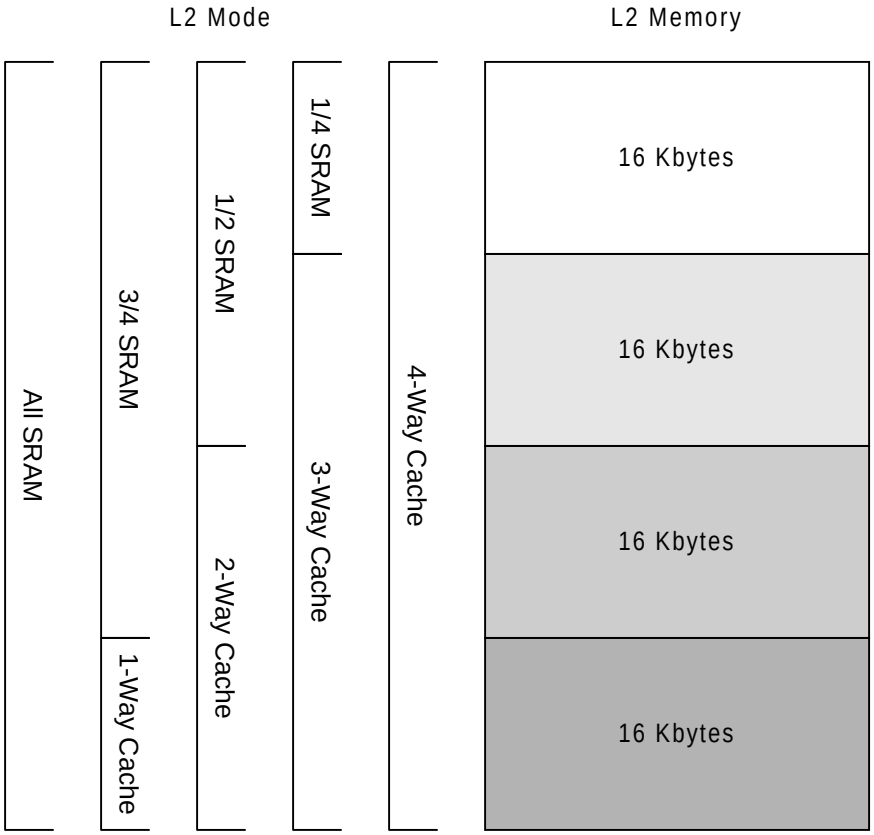
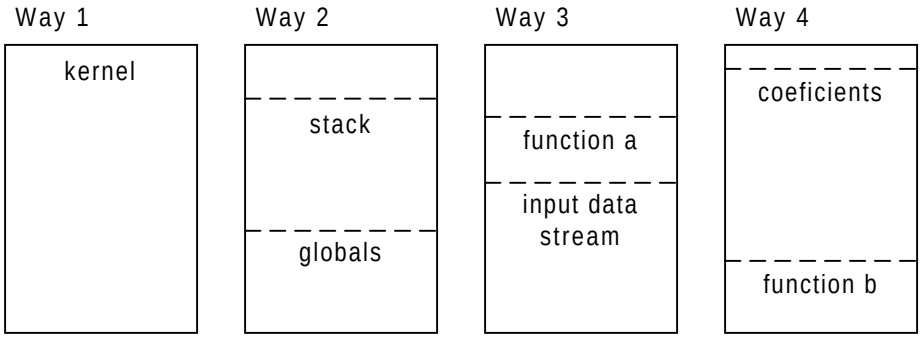


Figure 4. L2 Memory Configurations, Diagram 2



By providing a high level of associativity, the L2 minimizes thrashing between multiple data sources. For example, your application could execute program data on an array of coefficients, another data array, and a stack. The associativity of the L2 eliminates thrashing between the data since each data source can be cached by a different cache way. Figure 5 depicts an example of how multiple data streams can reside in the L2 without thrashing.

Figure 5. Data Allocation in Multiple Cache Ways



The L2 supports accesses from the L1P, the L1D, and the Enhanced DMA (EDMA). The EDMA can only access blocks of L2 memory that are configured as SRAM. The L2 memories are organized as four 64-bit wide banks. Two simultaneous accesses are serviced without stalling if the two accesses do not use the same bank. Thus, concurrent accesses by the L1D and EDMA busses to different banks are serviced without stalling.

Since the same memory location can be simultaneously cached in both the L1 and L2, the `C6211 must ensure that it is always accessing the current state of all memory locations. For example, if the CPU reads data that is not in L1D or L2 that data is fetched from external memory. The same data is then written into both the L1D and an L2 cache block. If the CPU then modifies that memory location by writing to its address, the data will only be updated in the L1D. In this case, the L1D is correct and the L2 contains the old data value. When this location is written out to external memory, for example to write an array of data to an external peripheral, the `C6211 must write the correct value. The TMS320C6211 uses snooping to ensure that the latest data is written any time that an L2 location is written back to external memory. The L2 snoops the L1 by checking if the memory location that is cached in the L2 is also cached in the L1. If so, the L1 informs the L2 if that memory location has been modified. If the data is in the L1 and has been modified then the L2 retrieves the modified data from the L1, sends that data to the external memory, and removes the data from both the L1 and the L2.

2 Cache Performance

The two-level cache of the TMS320C6211 achieves a high level of performance. Tests have shown that typical applications running on the TMS320C6211 operate at greater than 80% of the optimal cycle count performance of a `C62x device. Optimal performance would be achieved only by having infinite internal program and data memory.

Table 1. TMS320C6211 Benchmark Performance

Application	Efficiency
v.34	89%
AC-3 Decoder	90%
Zlib File Compression	96%
Line Echo Cancellation	99%
GSM Frame Encoder	92%
GSM Frame Decoder	88%
ADSL	85%

This level of performance is achievable due to the high hit rate of the L1 cache. Typically, 98% of program fetches execute without an L1P miss and 91% of data fetches hit in the L1D. When the L2 is configured as 4-way associative cache, normally 96% of the requests to the L2 are found in the L2 cache. Over 99.5% of all CPU cycles execute without requiring an access to external memory, virtually eliminating the access penalty associated with external memory devices.

2.1 Price / Performance

The primary focus of the TMS320C6211 is to achieve an easy to use, low cost, device with outstanding performance. The TMS320C6211 External Memory Interface (EMIF) has been optimized to operate a variety of devices. The ’C6211 offers 8-, 16-, and 32-bit interfaces to asynchronous memory, SDRAM, and SBSRAM devices. This enables you to take advantage of a high performance processor with single chip external memory solutions, reducing total system cost and board area.

2.2 Two-Level Cache Benefits

The cache architecture of the TMS320C6211 allows the device to achieve high performance without large amounts of expensive on-chip memory. By having an efficient cache, low cost, high-density external memory, such as SDRAM, is as effective as on-chip memory. Having a two-level cache provides several benefits over a one-level cache system. It allows reduced latency for an L1 cache miss, and it unifies the program and data in the same on-chip memory.

2.2.1 L2 reduces latency due to cache miss

By providing the L2 space, L1 cache misses may be serviced much more quickly. There is a significant reduction in cycle time for retrieving data from on-chip L2 memory than from an external memory. All external memory devices have significant startup latencies associated with them. By having the intermediate L2 cache, this latency is hidden from the user. The external memories that interface to the 'C6211 may operate at a maximum of 100 MHz, while the device operates at a 150 MHz maximum frequency. Using the fast L2 memories to cache the slower external memories reduces the latency of external accesses by a factor of five. The wide, high-bandwidth L2 bus transfers data at up to 1920 Mbytes/s while the EMIF interface operates at up to 400 Mbytes/s.

2.2.2 Unifying program and data in L2

By unifying the program and data in the L2 space, the L2 cache is more likely to hold the memory requested by the CPU. It enables the on-chip memory to contain more data than program when highly computational, looping code is being run to process large data streams. For long, serial code with few data accesses, the L2 may be more densely populated with program instructions. The unification allows you to allocate the appropriate amount of memory for both program and data and keeps the on-chip memory full of instructions and data that are the most likely to be requested by the CPU.

2.3 Real-time operation

An important concern in cache systems is that the device be able to perform in real time. There are several requirements for a system to ensure that real-time operation is possible. The operation of the device must be predictable, interrupts to the CPU must be handled without affecting the continued real-time operation of the device, and efficient I/O must be maintained.

2.3.1 Predictability

The TMS320C6211 has a high degree of predictability. Device operation typically achieves over 80% of performance of a `C62x device with infinite on-chip memory. Software tools to simulate the performance of the cache will be available in early 4Q98 to help you optimize system performance.

2.3.2 Interrupt Latency

Interrupt handling is an important part of DSP operation. It is crucial that the DSP be able to receive and handle interrupts while maintaining real-time operation. In typical applications, interrupt frequency has not increased in proportion to the increase in device operation frequency. As processing speeds have increased, latency requirements have not.

The TMS320C6211 is capable of servicing interrupts with a latency of a fraction of a microsecond when the service routine is located in external memory. By configuring the L2 memory blocks as memory-mapped SRAM, it is possible to lock critical program and data sections into internal memory. This is ideal for situations such as interrupts and OS task switching. By locking routines that need to be performed in minimal time, the microsecond delay for interrupts is reduced to tens of nanoseconds.

3 Efficient I/O Capability

Peripherals are a feature of most DSP systems that can take advantage of the memory-mapped L2 RAM. Typical processors require that peripheral data first be placed in external memory before it can be accessed by the CPU. The TMS320C6211 can maintain data buffers in on-chip memory, rather than in off-chip memory, providing a higher data throughput to peripherals. This increases performance when using on-chip McBSPs, the HPI, or external peripherals. The EDMA can be used to transfer data directly into mapped L2 space while the CPU processes the data. This increases performance since the CPU is not stalled while fetching data from slow external memory or directly from the peripheral. Using this method for transferring data also minimizes EMIF activity, which is crucial as data rates or the number of peripherals increase. Figure 6 illustrates the data flow from a peripheral to a typical processor. Figure 7 shows the same data flow from a peripheral to the TMS320C6211.

Figure 6. Typical processor peripheral data flow

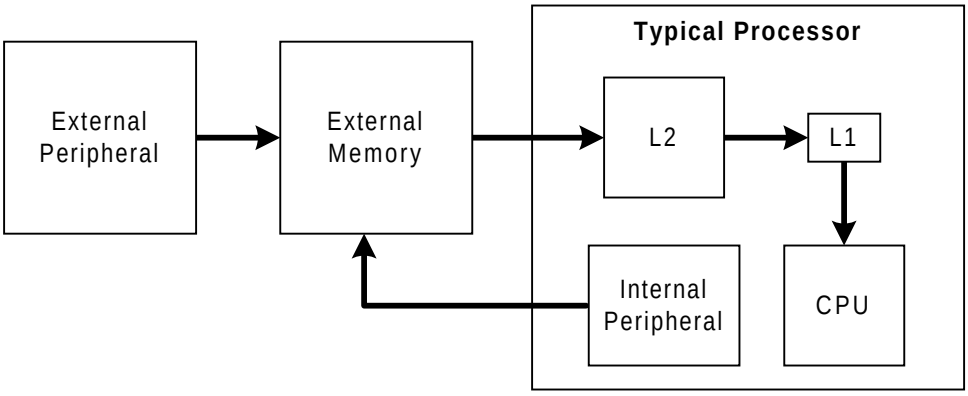
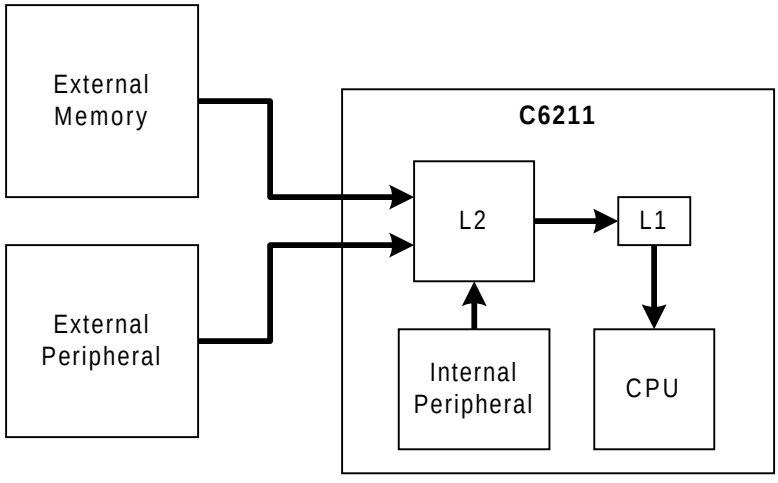


Figure 7. TMS320C6211 peripheral data flow



3.1 Ease of use

The efficiency of the ’C6211 cache architecture makes the device simple to use. The cache is inherently transparent to the user. Due to the level of associativity and the high cache hit rate, virtually no optimization must be done to achieve high performance. Reduced time for optimization leads to reduced development time, allowing functional systems to be up and running quickly. High performance can be immediately achieved with the ’C6211 cache architecture, while Harvard architecture with small internal memory requires much more time to achieve similar performance. This is because optimizing an application on a small Harvard architecture requires several iterations to tune the application to fit in the small, fixed internal memories.

4 Summary

The TMS320C6211 two-level cache architecture is optimized for DSP applications. The ’C6211’s caches deliver high performance without the cost of large arrays of on-chip memory. They provide better than 80% efficiency for all applications while greatly reducing development time. The ’C6211 utilizes an efficient low-cost cache architecture that will enable many new applications, such as low-cost DSL client modems and imaging, and provides the performance to take many existing applications to the next level.