

Data Cache Design Considerations for the Itanium® 2 Processor

Terry Lyon, Eric Delano
 Hewlett-Packard, Fort Collins, CO
 terry_lyon@hp.com
 eric_delano@hp.com

Cameron McNairy, Dean Mulla
 Intel Corporation, Santa Clara, CA
 cameron.mcnaury@intel.com
 dean.mulla@intel.com

Abstract

The second member in the Itanium Processor Family, the Itanium 2 processor, was designed to meet the challenge for high performance in today's technical and commercial server applications. The Itanium 2 processor's data cache microarchitecture provides abundant memory resources, low memory latencies and cache organizations tuned to for a variety of applications.

The data cache design provides four memory ports to support the many performance optimizations available in the EPIC design concepts, such as predication, speculation and explicit prefetching.

The three-level cache hierarchy provides a 16KB 1-cycle first level cache to support the moderate bandwidths needed by integer applications. The second level cache is 256KB with a relatively low latency and FP balanced bandwidth to support technical applications. The on-chip third level cache is 3MB and is designed to provide the low latency and the large size needed by commercial and technical applications.

1. Background

The Itanium processor [1] was the first implementation for the Itanium Processor Family instruction set architecture (ISA) based on the principles of EPIC (Explicitly Parallel Instruction Computing)[2]. The Itanium Processor architecture, as it is also called, was co-invented by Intel and Hewlett-Packard. The designers of the Itanium 2 processor, the second implementation of that ISA, were challenged to further enhance EPIC performance on high-end commercial and technical server applications. While the Itanium 2 processor includes performance features throughout the design, this paper will focus only on the data cache microarchitecture.

2. Data Cache Design Considerations

2.1. The Demands of a new Architecture

The demands of most contemporary superscalar and out-of-order microarchitectures are commonly supported with two memory pipelines (or M ports). The performance enhancement additions to the EPIC architecture and the wide issue width require increased accesses to the M ports and thereby put more pressure on the M resources [6]. The Itanium 2 processor provides 4 M-ports to support these demands. Dynamic Spec95 instruction counts for Itanium 2 processor-tuned code show that 30% of all instructions in the integer suite and 40% of the instructions in the floating-point suite are memory operations (not including M-type ALU operations or no-ops). These memory operations include normal loads and stores as well as explicit prefetches, predicated memory requests, speculative memory requests and their checks.. The 4 M-ports of the Itanium 2 processor provide headroom to cover peak activity periods as well as additional M-type ALU functions.

Instruction Bundling. The EPIC design concept has a fundamental instruction packaging, called a "bundle", which is made up of three instructions. A processor's issue width is normally defined as a multiple number of these bundles. For the Itanium 2 processor, the maximum issue width is 2 bundles, or six instructions, per processor cycle.

Table 1: Combination of templates for full 2-bundle execution requiring more than 2 M units.

	MII	MLX	MMI	MFI	MMF	MIB	MBB	BBB	MMB	MFB
MII			▲		▲				▲	
MLX			▲		▲				▲	
MMI	▲	▲	▲	▲	▲	▲	▲		▲	▲
MFI			▲		▲				▲	
MMF	▲	▲	▲	▲	▲	▲	▲		▲	▲
MIB			▲		▲				▲	
MBB			▲		▲				▲	
BBB										
MMB	▲	▲	▲	▲	▲	▲	▲		▲	▲
MFB			▲		▲				▲	

The instruction set specifies 24 template types which define the types of instructions within the bundle.

A two M port data cache can only support 52% of the template combinations for a full 2-bundle issue design as shown in Table 1. The McKinley four M-ports support full 2-bundle issue of all template combinations. (Note that the Itanium 2 processor has additional branch restrictions which may limit full issue of templates with B instructions.) Dynamic instruction counts of code tuned for the Itanium 2 processor show that in vortex 14% of the issued bundle pairs require 3 or more M-ports. In tomcatv that number rises to 30%. (These numbers include memory and ALU operations but not no-ops.)

Predication. Predication is a key feature of an EPIC design for reducing unpredictable branching at the cost of additional computation that may be “thrown away”. In some code segments, executing both paths of an unpredictable branch in parallel may be the only way to improve performance.

Code with predication can increase the number of memory operations requested, even though the operations were “predicated off”. In the in-order pipeline design of the Itanium processors, the predicate state for memory operations is not known precisely until the previous issue group has executed. As a result, a “predicated off” memory instruction will require the same issue and execution memory resources as would a “predicated on” instruction.

Control Speculation. EPIC control speculation allows loads to be executed speculatively without the penalty of faults or traps until the data is consumed. Speculation can lead to requests that will never be used, potentially increasing the demands on M-port resources. The check for a speculative load is required when the data is first non-speculatively used, and that check may require either an I or M type resource. Aggressive control speculation increases demand for M-port issue resources through extra speculative requests and additional check instructions.

Data Speculation. EPIC data speculation solves the problem of memory disambiguation by allowing loads to be advanced ahead of stores. An optional hardware structure called the Advanced Load Address Table (ALAT) detects when data accessed by an advanced load is invalid due to store updating the load’s contents. Advanced loads must be checked when they are used by issuing a load check (ld.c) or check ALAT (chk.a) instruction. These checks require M-port issue resources. Issuing advanced loads can improve performance by bringing in the data early, but the increased performance comes at the cost of doubling the demand on M-port resources. Use of data speculation varies with different compilers and optimization levels, but on jpeg code tuned for the Itanium 2 processor the percent of loads using data speculation was 28%.

2.2. Demands of an Optimizing Compiler

A great deal of the performance of any Itanium family processor comes from the compiler and its ability to make the best use of the available execution resources. Itanium instruction set compilers must deal with the challenges of bundles, predication, data speculation, control speculation, instruction and data prefetching, and hinting for data allocation and branch prediction. All these techniques must then be matched with the capabilities of a specific implementation. One way the processor designer can help compilers and also improve performance is by reducing hardware restrictions visible to the compiler.

The Itanium 2 processor’s “optimal 2 banger” gives more flexibility to the compiler because most of the 2 bundle template combinations are now fully executable. Many bundling and instruction scheduling restrictions are eased by providing 4 M-ports that can execute both memory and ALU operations.

Many contemporary processors have load latencies greater than 1 cycle for their first level cache. In an out-of-order processor design, the latency can be somewhat mitigated by deep issue queues, but in the in-order execution implementation of the Itanium 2 processor, the first level cache latency becomes a difficult issue for the compiler to manage. To achieve the highest performance possible for a 2-bundle issue processor, a first level cache load latency of 2 cycles may require the compiler to schedule the load up to 11 instructions ahead of its use. In some cases this is possible, but in other cases performance is degraded through hardware imposed pipe stalls or explicit stop bits, each causing reduced instruction execution. Any stall for a wide-issue processor can be costly for performance since it affects all instructions in an issue group. The Itanium 2 processor provides a 1-cycle first level data cache to enable compilers to better realize the performance made possible by hardware capable of issuing 6 instructions per cycle.

2.3. Demands of Customer Applications

Commercial Applications. Unlike many benchmarks that show high utilization of local caches, commercial database applications, and in particular TPCC workloads, place extreme demands on a processor’s memory subsystems. Instruction footprints of such applications can only be contained in a very large cache. Data references show little locality resulting in many memory references that cannot be satisfied by any cache level [4]. These applications are *latency sensitive* [5]. Decreasing memory latency is a good way to improve performance. The Itanium 2 processor data cache was structured to enable fast access even to the largest level of cache.

Technical Applications. Technical applications have many of the same characteristics of commercial workloads, in that they may need to reside in the largest levels of cache. Simulation codes may require streaming of instructions from a large cache, while data-crunching code may require large streams of data to be processed. These applications tend to be *bandwidth sensitive*.

The Itanium 2 processor data cache design sought to support high-end technical applications by providing the capability to execute a stream of two MMF bundles per cycle, requiring a wide datapath from the lowest level floating point cache all the way to the system bus.

3. Data Cache Microarchitecture

3.1. 1-cycle Integer Load WITH 4 Memory Ports

An important achievement in the Itanium 2 microprocessor is the ability to provide a 1-cycle latency (zero load use) integer load *in combination with* a four-ported data cache. This four-ported data cache is able to execute four M operations per cycle. Figure 1 illustrates the address flow of the L1 cache as it fits within the overall data cache input stages.

Several microarchitectural techniques enable the 1-cycle L1 data cache *and* the four-port data cache:

- L1 services only integer loads and stores
- Restricting the load ports to two of four M ports
- Restricting the L1 cache to a physical cache index
- Providing a dedicated, but smaller, integer load TLB
- Intimately integrated “prevalidated” cache tag, dedicated only for integer loads

The difficulty in achieving the 1-cycle cache for the Itanium 2 processor running at 1 GHz in 0.18-micron technology is partially illustrated by the fact that the original Itanium processor at 800 MHz required 2 cycles for the same sized cache with the same process technology.

A primary microarchitectural enabler for the 1-cycle

cache access is the restriction of the L1 data to be used *only for integer loads*. A first benefit of this restriction is related to the physical connections in and out of the L1 data cache. Memory addressing is sourced from the integer register file and the restriction that L1 provides only integer load data allows a very tight physical positioning of the L1 with the integer register file and data path. This restriction allows the data cache to be smaller, to be supported by a smaller TLB (since floating point and store pages are not allocated in the TLB), and to have a reduced bandwidth of two load operations per cycle.

A distinction of the Itanium 2 processor L1 cache is the separation of memory ports and their functionality. Memory ports 0 and 1 support integer load operations, while ports 2 and 3 support integer stores. The L1 cache supports a full streaming of two loads *and* two stores per cycle.

The capability for compilers to schedule 2 integer loads and 2 integer stores per cycle is a powerful tool for code optimizations. The compiler can schedule memory operations in a convenient order. The dispersal hardware will route memory instructions to the appropriate M-port and will break the bundle pair if too many M resources are required.

Important for the fast L1 access time is a restriction on the cache organization to enable physical indexing of the cache memory and tags. For the Itanium 2 processor, the decision was made to limit the L1 to a physically indexed 4-way 16 KB cache.

Associated with the L1 cache is a small and dedicated translation look-aside buffer (L1-TLB). In the Itanium 2 processor this buffer holds 32 entries with 4 KB pages. This TLB is small but it is used only for integer load accesses and not for accessing the much larger L2 or L3 caches or for stores. Another simplification is that this TLB does not hold protection information for the memory page—protection information is stored in a larger and parallel TLB structure called the DTLB (Data TLB). The DTLB provides variable page size support, protection control and the required architectural functionality.

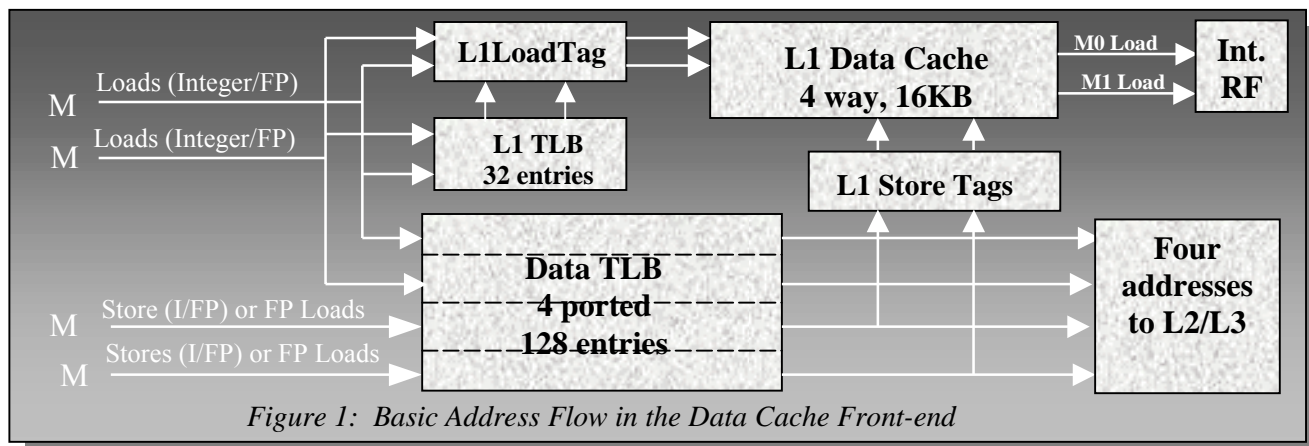


Figure 1: Basic Address Flow in the Data Cache Front-end

Putting all of this capability in the L1-TLB would make the 1-cycle integer data cache access more difficult. Using this dual-TLB structure requires “hits” in both the L1-TLB as well as the DTLB to enable an integer load data cache hit, other instructions need only a DTLB translation hit.

A final microarchitectural technique to enable the 1-cycle cache is a prevalidated cache tag, used specifically for L1 integer loads. This prevalidated cache tag structure is detailed elsewhere [3]. A prevalidated cache tag provides a fast way hit signal for the L1 data cache lookup. Stores and invalidations are received on memory ports 2 and 3, and are translated through the larger DTLB structure and through a traditional cache tag for the L1. Using the DTLB for the integer stores provides a unified path for all stores and cache invalidation operations. Floating-point stores use this same path to invalidate cache data in the L1 cache. Separating out the integer stores, floating-point stores, and invalidation operations form the L1-TLB and the L1 integer load cache tags, further enables an optimized 1-cycle cache access.

3.2. Three levels of Cache AND fast access to the Largest Caches

High-end server applications require large caches and access them frequently. For technical applications the data streaming bandwidth is generally more important than access times, but for some commercial applications (like TPC-C) the latency to these large caches has significant performance implications. The Itanium 2

processor data cache microarchitecture provides three-levels of cache, tuned to support a variety of applications, with fast access to all levels of cache, and even to the system bus for cache misses.

Several microarchitectural techniques used in the Itanium 2 processor 3-level data cache structure are illustrated in Figure 2:

- 3-level cache, tuned for integer and FP applications
- L2 accesses start in parallel with the L1 access
- Fast L3 accesses are enabled by an early L2 tag access

The Itanium 2 microprocessor has implemented a 16 KB store write-through L1 cache that is closely integrated with the integer register file, and is used to supply integer load data in 1 cycle (0 load use). The L2 cache handles all other memory instructions—integer stores and floating-point loads and stores since they are less sensitive to latency and require larger caches. The 256KB L2 provides fast access times: 5 cycles for integer and 6 cycles for floating-point loads. L2 is pseudo 4 ported through the use of 16 banks of memory. The output of each L2 bank provides enough data to load an extended floating-point operand or a floating-point load pair instruction. The Itanium 2 processor’s largest cache is the on-chip 3MB L3 that has a minimum access time of 12 cycles. The Itanium 2 processor provides fast access to all levels of cache.

Critical to improving the access times to the L2 cache (and thus also the L3) is that the L2 access starts in parallel with the L1 access (see figure 2). All memory instructions for all four M ports begin by accessing the

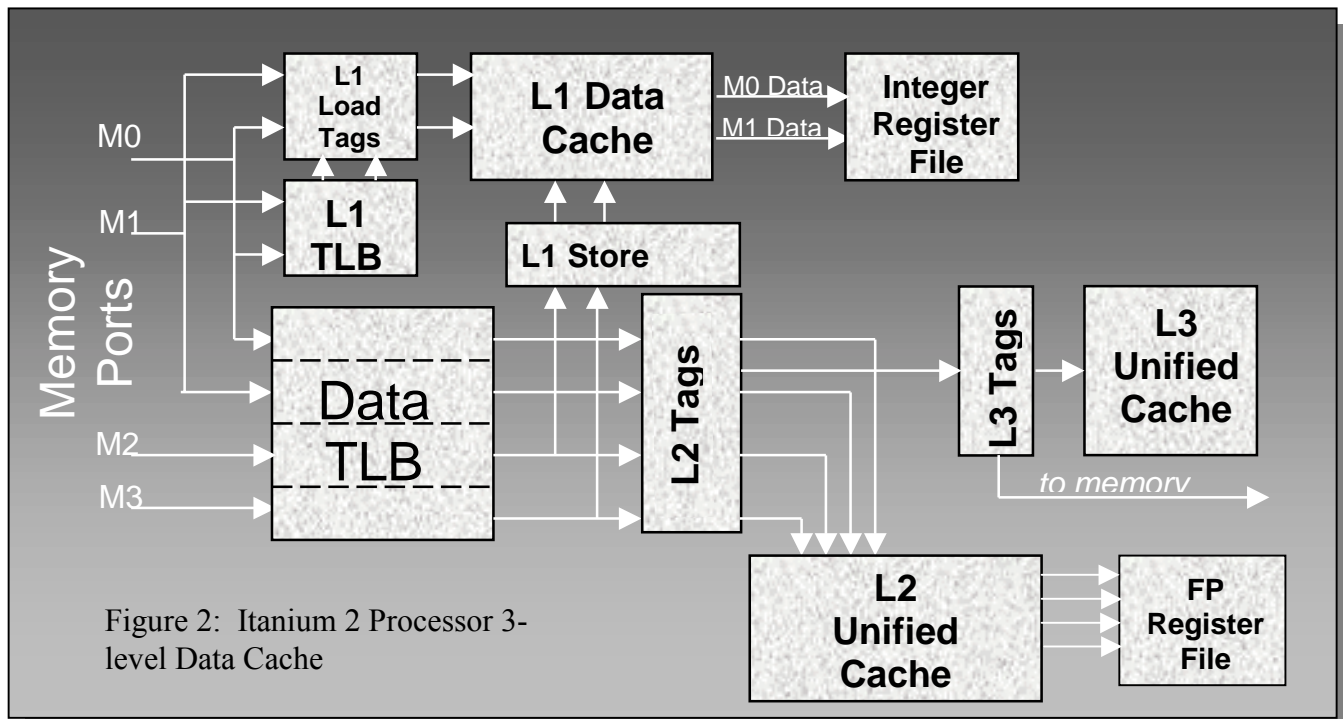


Figure 2: Itanium 2 Processor 3-level Data Cache

DTLB (data TLB) and also in parallel, indexing into the L2 tags during the same cycle that L1 is being accessed. The four memory addresses coming from the DTLB begin the L2 tag compare immediately at the end of the first data cache cycle. These memory addresses go to the L2 tag unconditionally, without any regard to the type of instruction or even the presence of an L1 hit. The L2 tag access and L2 hit compares are completed in the second data cache cycle. In the middle of the second data cache cycle, an L1 miss can be speculatively forwarded to the L3 cache. Reading the L2 tags early prevents accessing the L2 data array when there is no L2 hit.

The DTLB is fully four-ported, fully associative and its access is completed in less than one cycle. The DTLB provides variable page sizes from 4KB up to 4GB, to allow mapping of very large caches and even larger datasets.

Since the Itanium 2 processor pipeline is in-order, it is important to note that all data caches are non-blocking. At least 48 memory operations can be outstanding throughout the memory system without stalling the main pipeline. Out-of-order memory access begins at the L2 and continues throughout the memory hierarchy. The 32 entry L2 memory instruction buffer re-issues memory requests to the L2 data array taking into account all banking and ordering restrictions.

3.3. Real Floating Point Data Streaming Support

The Itanium 2 microprocessor data cache provides the memory streaming support required to truly match the high bandwidth of the two floating-point multiply-add units and large floating-point register file provided on the Itanium 2 processor. This memory streaming capability is also useful for other operations such as block or page copies from memory to memory. The Itanium 2 processor implemented several microarchitectural features

to optimize 2-bundle floating-point execution:

- Full double-bundle MMF template support
- High bandwidth data paths to and from the L2 cache and the FP register file
- High data streaming bandwidth between the L3 and L2 caches
- Large cache line sizes on the system bus, L3 and L2
- High bandwidth and increased frequency on the system bus

The Itanium 2 microprocessor design supports executing two MMF bundles per cycle in the 4 M-ports, providing very powerful floating-point streaming performance. This capability is extremely valuable for codes such as the DAXPY kernel that is found in the Linpack benchmark. DAXPY can be executed on the Itanium 2 microprocessor with two load FP pair instructions, two FMAC operations and 2 FP stores per cycle. The hardware can deliver 4 floating-point operands from L2 to the floating-point register file AND receive and process 2 stores per cycle. This is illustrated in Figure 3.

Many “blocked” applications can fit into the L2 cache, but a larger L3 cache nearby is sometimes needed for periodic “refueling” of the L2. The Itanium 2 processor data cache microarchitecture provides data paths between the L2 and L3 caches which are both 32 bytes in size. These data paths provide a maximum bandwidth of 4 floating-point operands to be transferred from the L3 to the L2 each cycle. In addition to the load bandwidth, the writeback path from L2 to L3 can also support a maximum of 4 floating-point operands transferred per cycle in parallel with the load-to-L2 operations. The “refueling” bandwidth between the L3 and L2 matches the execution bandwidth of the dual floating-point multiply-add units of the Itanium 2 processor.

Automatic prefetching is provided by the large 128B cache line size on the Itanium 2 processor. Using explicit

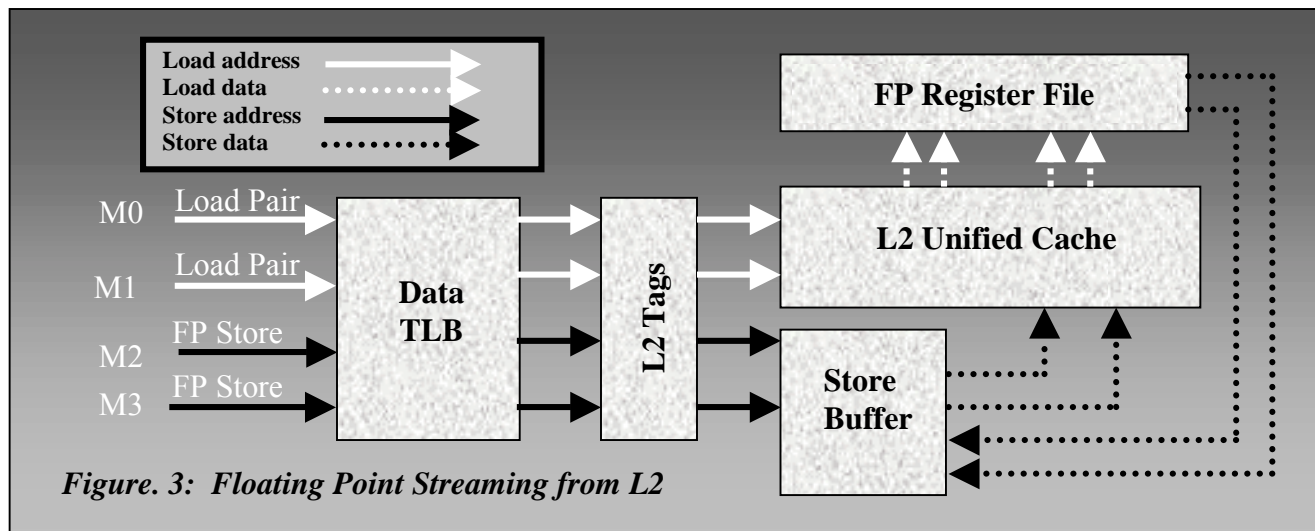


Figure. 3: Floating Point Streaming from L2

data prefetching instructions can cost M-port resources and increases data cache queue activity. Increasing the cache line size provides an effective way to implement data prefetching without the costs of explicit prefetching.

A final component to the “refueling” of the Itanium 2 microprocessor’s data cache is that of efficiently bringing data into the processor from the system memory. To that end, the Itanium 2 processor has optimized the front-side bus data width and frequency to provide a maximum of 6.4 GB/sec data transfer rate over the system bus.

3. Results

The one cycle L1 data cache was a major design accomplishment and it has a significant impact on the Itanium 2 processor performance. In Table 2 the performance gain for a 1-cycle L1d cache (vs. a 2-cycle cache of the same size) is indicated for the SpecInt2000 benchmarks. For these simulations, which were run on the Itanium 2 processor, the compiler was forced to schedule for a 2-cycle data cache to provide the 2 cycle cache data point. In addition to the performance loss for a 2-cycle cache, the instructions executed for these benchmarks increased from 1-4% over code optimized for a single-cycle data cache. Although not discussed in this paper another benefit of this 1-cycle cache design is that it is also used in the L1 instruction cache, and there it provides the benefit of a 0-cycle branch re-steer and a reduced penalty for a miss-predicted branch re-steer.

The Itanium 2 processor has many microarchitectural enhancements to support floating-point technical applications. Table 3 provides results from the University of Utah mathematics department benchmark suite [7]. The results compare the original Itanium processor at 800 MHz and the Itanium 2 processor at 1 GHz. The best compiled results for each processor and each benchmark are reported. These benchmarks are meant to reflect real scientific and engineering applications. The results show an improvement of nearly 2x performance for the Itanium 2 processor over the original Itanium processor.

A third feature of the Itanium 2 processor data cache microarchitecture is the low latency L3. Table 4 illustrates the penalty for increased L3 latency on transaction processing performance. The baseline is the performance for the 12-cycle L3 latency provided by the Itanium 2 processor.

4. Conclusions

The Itanium 2 microprocessor data cache architecture sets a new standard for high-performance processor designs. A single cycle integer L1 cache, a high-bandwidth floating-point data streaming L2 cache, and a large, but fast, on-chip L3 cache provides the structure for the data cache. The 4 memory ports provide compilers

the resources to effectively optimize for a full execution of 6 instructions per cycle. By implementing a parallel access of L1 and L2, and by accessing the L2 tag ahead of the L2 data, the access times to the L2 and L3 caches can be made amazingly fast.

Specifications for the Itanium 2 processor data caches are given in Tables 5.

5. References

[1] Harsh Sharangpani, et. al., “Itanium Processor Microarchitecture,” *IEEE Micro*, Sept-Oct 2000, pp. 24-43. Also “Itanium Processor Microarchitecture Reference for Software Optimization”, August 2000, Order Number 245473, Intel Corp or <http://developer.intel.com/design/ia64/downloads/245474.htm>.
 [2] M.S. Schlansker and B. R. Rau, “EPIC: Explicitly Parallel Instruction Computing,” *Computer*, Feb 2000, pp. 37-45. Also “Intel IA-64 Architecture Software Developer’s Manual”, Order # 245317-9 (volume 1-4), Intel Corp.
 [3] D. Bradley, P. Mahoney, and B. Stackhouse, “The 16kB Single-Cycle Read Access Cache on a Next Generation 64b Itanium Microprocessor”, *2002 IEEE International Solid-State Circuits Conference Digest of Technical Papers*, pages 110-111, Feb., 2002.
 [4] A. Maynard, C. Donnelly, and B. Olszewski, “Contrasting Characteristics and Cache Performance of Technical and Multi-User Commercial Workloads”, *Proc. Intl. Conf. Arch. Support for Prog. Lang. And Op. Sys.*, pages 145-156, Oct. 1994.
 [5] L. Barroso et. al., Impact of chip-level integrations on performance of OLTP workloads”, *Sixth Intl. Sym. On High-Performance Computer Architecture*, pages 3-14.
 [6] S. Cho, P. Yew, G. Lee, “A high-bandwidth memory pipeline for wide issue processors”, *IEEE Transactions on Computers*, Vol. 50, Issue 7, July 2001, pages 709-723.
 [7] <http://www.math.utah.edu:8080/pub/benchmarks/local/index.html>

Table 4: Performance loss for added L3 latency

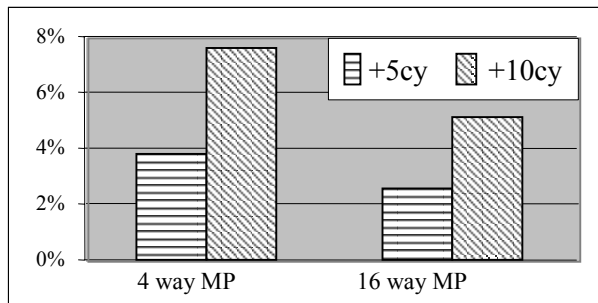


Table 2: Performance gain for a 1-cycle L1d cache

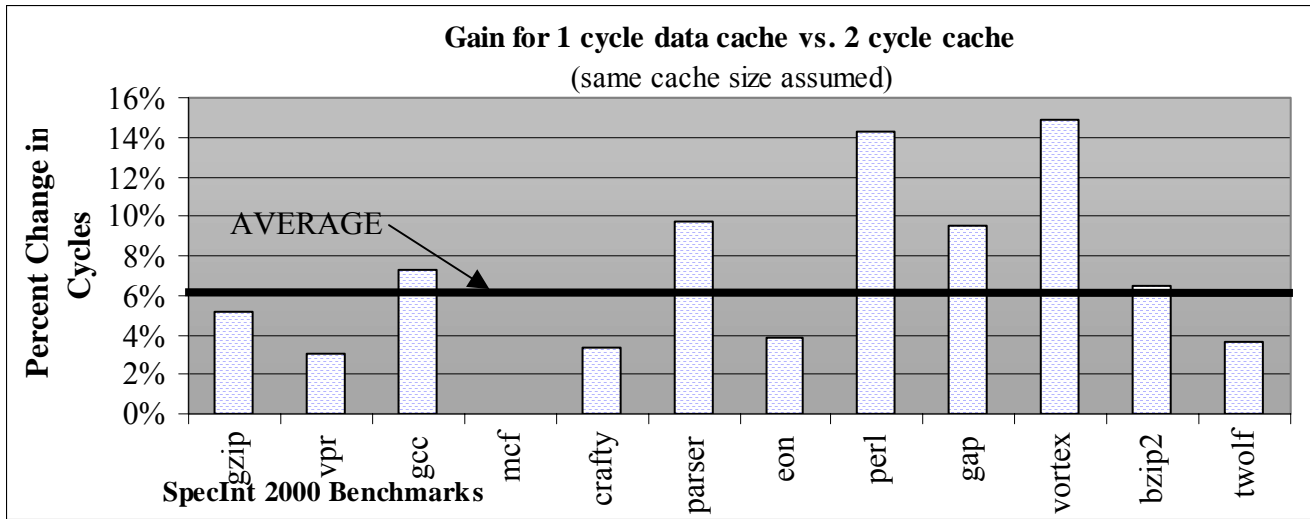


Table 3: Performance on floating-point intensive applications

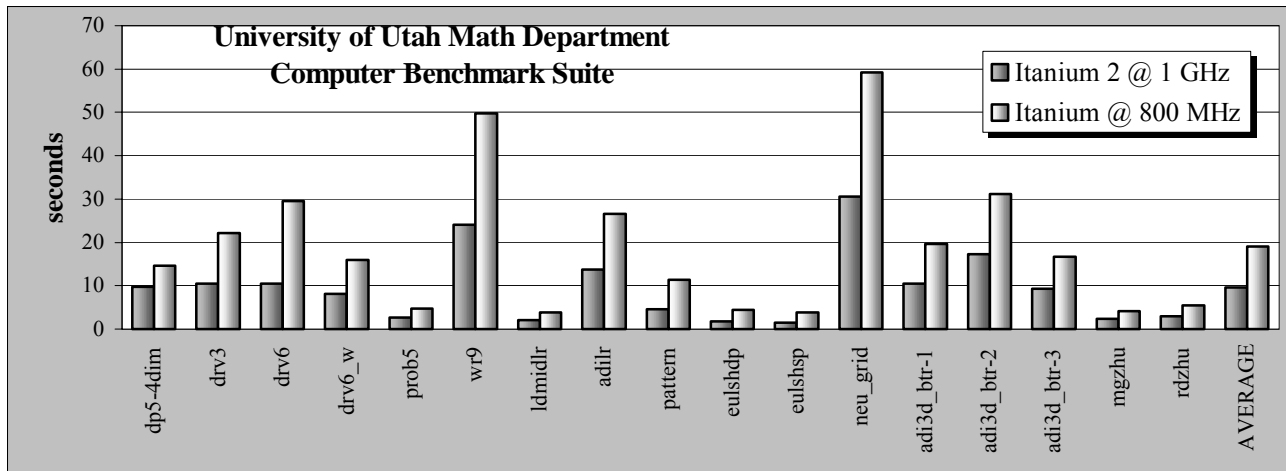


Table 5: Itanium 2 Processor Data Cache Specifications

	L1D	L2	L3
CONTENT:	Data Only	Unified: I + D	Unified: I + D
SIZE:	16 KB	256 KB	3 MB
ASSOCIATIVITY:	4 way	8 way	12 way
MIN. LATENCY:	1 cycle (integer)	5 (Int), 6 (FP), 7 (instr)	12 (Int), 13 (FP), 14 (instr)
LINE SIZE:	64 bytes	128 bytes	128 bytes
ORGANIZATION:	8 "groups"	16 banks	1 bank
MEMORY PORTS:	2 read + 2 write	4 read or write	1 read or write
READ DATA PATH WIDTH:	2 x 64 bits	4 x 82 b (FP) + 2 x 64 bits	1 x 32B (to L2)
WRITE DATA PATH WIDTH:	2 x 64 bits	(int)	1 x 32B (from L2)
TAG DESIGN:	Prevalidated Load Tag, Conventional Store Tag	2 x 82 bits Conventional Tag, early tag	Conventional Tag
TAG PORTS:	2 read + 2 write	4 read or write	1 read or write
STORE POLICY:	Write-through	Write-back	Write-back
CACHE ALLOCATION POLICY:	On cacheable integer load miss or prefetch	On any cacheable L2 miss	On any cacheable L3 Miss
FILL RATE:	32 bytes/cycle	32 bytes/cycle	32 bytes/cycle

