



White Paper
Joshua Ruggiero
Computer Systems Engineer
Intel Corporation

Measuring Cache and Memory Latency and CPU to Memory Bandwidth

For use with Intel[®]
Architecture

December 2008



Executive Summary

With today's computer architecture industry focused on clock speed and processing core count, it is easy to lose sight in other factors that play important roles in overall system performance. Memory latency and bandwidth are two major factors of performance. These two key aspects of system architecture can be broken up into processor cache and system memory. The open source LMBench Benchmarking suite provides utilities to accurately measure latency and bandwidth.

[Latency and Bandwidth are important measurements to take.](#)

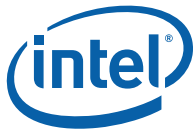
Latency and Bandwidth are important measurements to take. These measurements are good indicators that describe the impact during the tuning processes, compare data from one platform to another and possibly identify other potential performance areas of improvement. The open source model provides a solution that adds no extra cost of benchmarking and portable measurements between customers and manufacturer, different teams within the same organization, or even separate members of the same team.

§



Contents

Background	4
Measuring Latency and Bandwidth.....	7
Introduction to the Tools	7
How to take Measurements	8
Understanding the Results	10
Conclusion	12



Background

In an industry that is so dependent on performance, it is easy to focus on speed of the CPU or the number of cores per socket. However, memory performance is equally as critical to total system performance. If a CPU is not properly matched with correct memory, it is possible for the CPU go underutilized, causing decreased efficiency and poor overall performance. There are two key factors of performance by which a computer system retrieves data. Latency and bandwidth play large roles in system performance and data manipulation.

In the Intel® architecture three-chip design, the CPU connects to the Memory Controller Hub (MCH) through a bus call the system bus or Front Side Bus (FSB). The MCH then connects to the I/O Controller Hub (ICH) through a link called Enterprise South Bridge Interface (ESI). The block diagram in [Figure 1](#) shows an example of these connections with a dual processor configuration (referred to as P1 and P2) connected to the Intel® 5100 MCH.

The CPU will request data by issuing a memory read on the FSB. This read travels from the CPU, through the FSB, into the MCH and over to the memory location as determined by the memory controller. Latency refers to the time it takes for this request to be processed and for the first portion of data to arrive back at the requesting agent, in this case, the CPU. Latency refers to many different paths within computer architecture, including reading from I/O devices. This paper only discusses latency in terms of CPU to memory and CPU to cache accesses. The methodology used in determining latency is the same for both Three Chip and SoC design architectures.

Intel also supports a System-on-a-Chip (SoC) design. In this design, the classic 3 Chip architecture becomes integrated into a single package. As shown in [Figure 2](#), the CPU continues connecting the memory controller through the Front Side Bus.

For the purpose of this paper, bandwidth will be considered the measured amount of data that passes through any given point per unit of time. Represented as a rate, bandwidth can also be discussed as throughput and given in the units of Megabytes per second (MB/sec). Megabyte is defined as 2^{20} or 1,048,576 bytes.



Figure 1. Intel® 3-Chip design with Multiprocessor

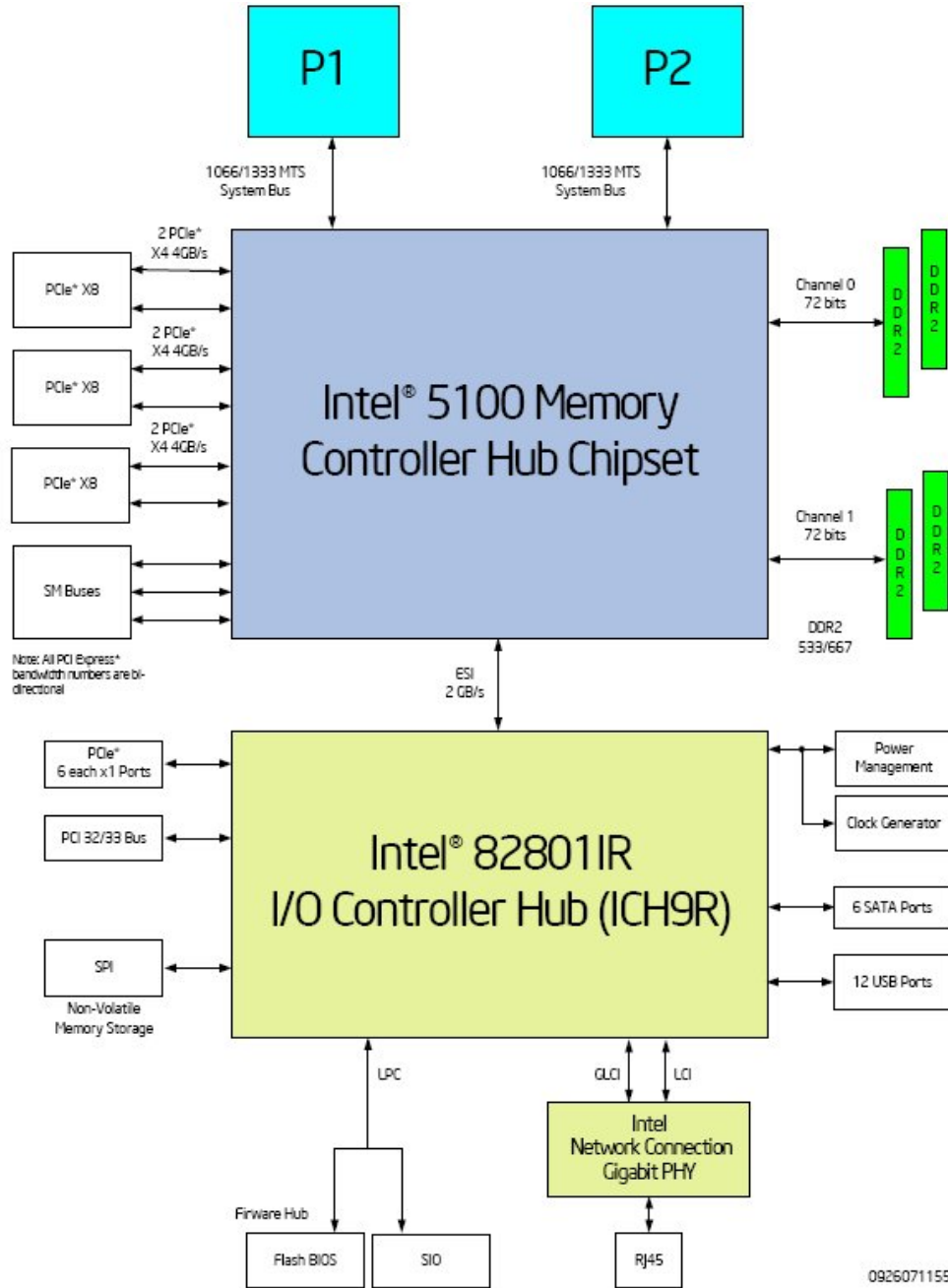
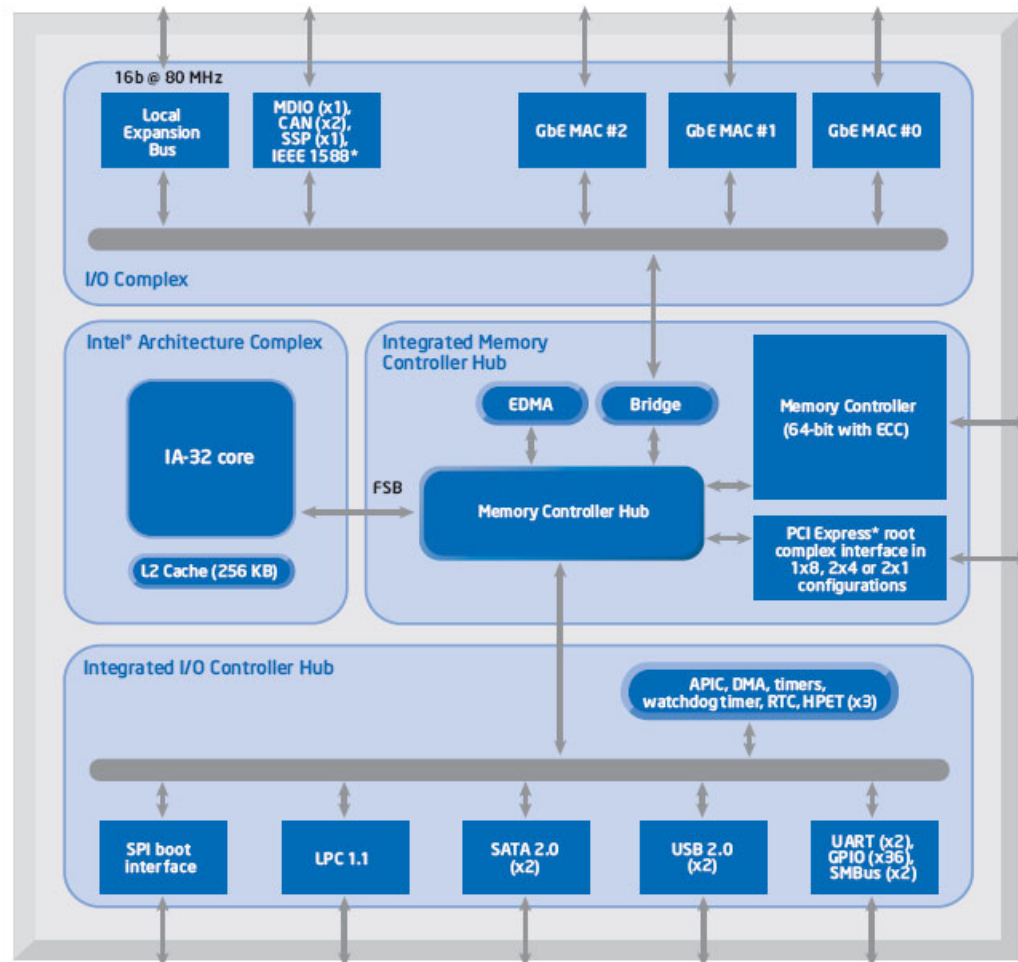




Figure 2. Block Diagram for the Intel® EP80579 Integrated Processor

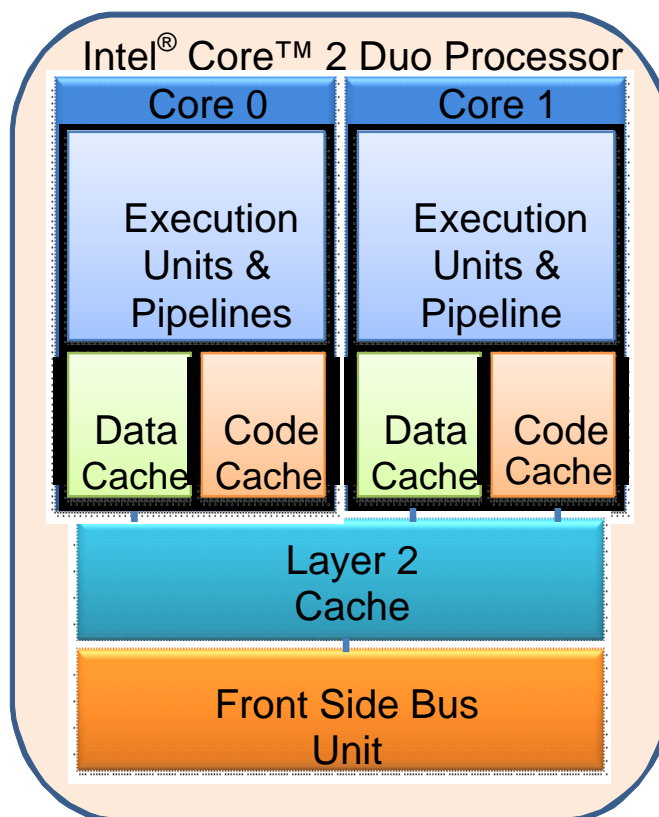


Numerous factors affect latency and bandwidth. Intel® architecture uses a hierarchy to organize the memory structure. With the current architecture, in the instruction and data caches reside at the highest level. Generally a small amount and referred to as Level 1 (L1) Cache, this region resides at the core level. A larger shared section between cores referred to as Level 2 (L2) cache, has slower access times than the higher L1 cache. L2 cache resides in the same package as the cores and is shared among them. Going beyond cache is system memory. Because system memory, or DRAM, resides outside the processor, along the Front Side Bus, and into the memory controller hub, this will have a greater latency than the local caches inside the package of the CPU. Refer to [Figure 3](#) for a block diagram of the layout inside an Intel®



Core™ 2 Duo processor. The L1 cache is divided between data and instruction (code) caches and they are typically of equal size.

Figure 3. Simple block diagram of major components of the Intel® Core™ 2 Duo Processor



Measuring Latency and Bandwidth

Introduction to the Tools

In order to get a better understanding of the system performance, software measures latency and bandwidth. By utilizing tools from the open source community, measurements can be taken and then reproduced with no added cost, thereby making benchmarking portable across all parties involved on a project. Using open source Linux in combination with these utilities, a team can present easily reproducible data without the hassle of proprietary software or other tools.

There is a specific suite of open source utilities that measures cache latency, memory latency, cache bandwidth and memory bandwidth. This suite is



called LMBench. LMBench contains many utilities for different benchmarks, but two in particular are of concern when taking these measurements.

Background information on the tools used can be found here:

<http://www.bitmover.com/lmbench>

The following LMBench suite can be found here:

<http://www.bitmover.com/lmbench/lmbench3.tar.gz>

It is important to measure the whole data path. If a utility measured latency to local processor cache only, this would provide just a portion of the picture. With the "lat_mem_rd" utility from LMBench, it is possible to get accurate measurements. The utility measures all levels of cache and into main memory as specified by the commands passed to it.

It is also important to measure the bandwidth at the different levels as well. A utility from LMBench called "bw_mem" takes these measurements.

How to take Measurements

Measuring Latency. Download and unzip the LMBench utility. Install it by executing the make file. Use the executable binary file called "lat_mem_rd" found in the "bin" folder of the utility's directory. Next, use the following command line:

```
># taskset 0x1 ./lat_mem_rd -N [x] -P [y] [depth] [stride]
```

Where [x] equals the number of times the process is run before reporting latency. Typically setting this to '1' is sufficient for accurate measurements. For the '-P' option, [y] equals the number of processes invoked to run the benchmark. The recommendation for this is always '1.' It is sufficient to measure the access latency with only one processing core or thread. The [depth] specification indicates how far into memory the utility will measure. In order to ensure an accurate measurement, specify an amount that will go far enough beyond the cache so that it does not factor in latency measurements. Finally, [stride] is the skipped amount of memory before the next access. If [stride] is not large enough, modern processors have the ability to prefetch the data, thus providing artificial latencies for system memory region. If a stride it too large, the utility will not report correct latencies as it will be skipping past measured intervals. The utility will default to 128 bytes when [stride] is not specified. Refer to [Figure 4](#) for results graph. Binding to the first core and accessing only up to 256M of RAM, the command line would look as follows:

```
taskset 0x1 ./lat_mem_rd -N 1 -P 1 256M 512
```

This will run through each level of memory and report the latency in nanoseconds. Applying the "taskset" feature of Linux in a multi-core environment ensures that this utility use the same processing core for each individual run. If numbers are unrealistic, increase the stride of the command to avoid CPU prefetchers artificially inflating the data. Stride should be between 128 and 512. Depth can be anywhere between 64 and 256, ensuring



the utility goes deep enough into memory to not be using cache to affect the result.

Measuring Bandwidth. In order to measure bandwidth, LMBench provides a utility called "bw_mem." The following command line measures bandwidth:

```
># ./bw_mem -N [x] -P [y] [size] [traffic type]
```

The command line structure is similar to the latency utility where [x] is the number of times the process runs before reporting the result. The parameter [y] specifies the number of running processes used to measure the memory access, and is more critical with this utility. The [size] parameter allocates the specified amount of memory used for the measurement. The size used for this will double, due to the utility allocating twice the specified amount and zeroing it out to do the operations. Finally, the [traffic type] parameter specifies read, write or a combination of both read and write, from CPU to memory. The following are the options available; rd, wr, rdwr. Other operations are also available, but not discussed.

When measuring bandwidth of system memory in a multi-core or multi-threaded processor, adjusting the [y] parameter completely exercises the system memory. The number of processes specified with the [y] parameter multiplies the amount specified by the [size] parameter. So, when measuring 1024M of memory, for example, and using four cores to measure, [y] should be specified as '4' and [size] as 256M. If measuring cache bandwidth, take note of the memory hierarchy. Shared cache, such as L2 on the Intel® Xeon® 5400 series processors, behaves similarly to system memory. So as not to go beyond the amount of cache, take care in dividing shared cache among the number of processes, remembering twice the amount specified will be used for memory allocation. When accessing unshared cache, the [size] parameter will not change with the number of processes specified. With unshared cache, results will scale with number of cores used. Take caution when using the "taskset" function here. When running multi-core measurements, the possibility of isolating particular cores may be useful.

For example, when measuring latency bandwidth on the Quad-Core Intel® Xeon® Processor E5440, the L1 (data) cache size is 32 KB per core, L2 cache is 12 MB shared among all cores. L1 cache is divided up between data and instruction. For the E5440, total cache is 32 (data) + 32 (instruction), quad core makes this a total of 256 KB of L1 cache. Since each core has 32 KB of L1 data cache, and "bw_mem" uses twice the amount specified to take its measurements, only measure 16 KB for each core. The command line for measuring L1 reads looks as follows:

```
./bw_mem -N 10 -P 4 16384 rd
```

where 16384 is 16 KB, which is half the amount available in L1 data cache.

Now, to measure the L2 cache, the E5440 has a shared 12 MB available. With the utility using twice as much specified, max total should be 6 MB. Distribute



this between cores, giving 1.5 MB or 1,536 KB per core. The read bandwidth command line looks as follows:

```
./bw_mem -N 10 -P 4 1536K rd
```

In order to measure system memory, the command line is very similar to measuring L2, because system memory is also shared. If a total of 2 GB of memory is available, then half of this amount becomes distributed between all cores. For a quad core processor, the amount specified would need to be 256 MB. The command line is:

```
./bw_mem -N 10 -P 4 256M rd
```

Understanding the Results

Viewing the output of the latency measurements shows the first column where in memory the measurement happened and the second column is the latency figure presented in nanoseconds. The bandwidth measurements show a similar fashioned output, but with only one row of two columns. The first figure is the size, in MB, and the second is the bandwidth figured in MB/sec.

Latency Results. As seen in [Figure 4](#), graphing this data, the different levels of memory hierarchy become apparent. The Y-axis represents the latency in ns and the X-axis presented as memory depth in megabytes (MB). Since L1 and L2 cache latency ties to the core clock, CPU frequency plays a role in how fast memory accesses happen in real time. This means the number of core clocks stays the same independent of the core frequency. For a comparable result, it is best to convert the latency given by LMBench from nanoseconds into CPU clocks. To do this, multiply the latency by the processor frequency.

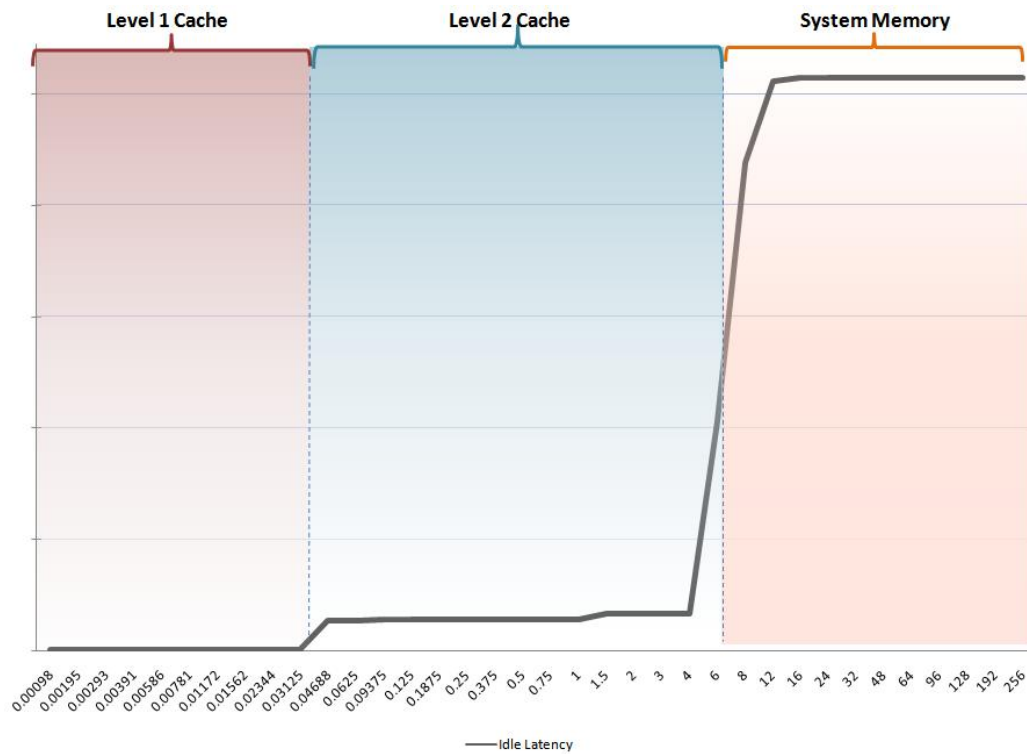
$$\text{Time(seconds)} * \text{Frequency(Hz)} = \text{Clocks of latency}$$

Therefore, if a 2.4 GHz processor takes 17 ns to access a certain level of cache, this converts to:

$$17 \times 10^{-18} \text{ seconds} * 2400000000 \text{ Hz} = 17 \text{ ns} * 2.4 \text{ GHz} \approx 41 \text{ Clocks}$$



Figure 4. Graphed Memory Latency



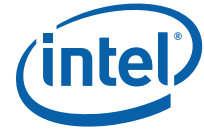
Bandwidth Results. Bandwidth results give the result in MB/sec. It is also important to note that when running write traffic, LMBench only measures the write portion and does not take into account reads for ownership and snoop traffic that also occur during a write. If cache and memory region is unknown, use the measurements from latency to decipher the different regions of memory.

Parameters that affect Results. Other factors that affect these results are CPU frequency, Front Side Bus speed, number of cores, speed of DRAM and number of DRAM channels, DRAM Ranks and CPU prefetchers. All these can change the outcome of the benchmarks discussed. When comparing results, ensure these parameters are the same. For example, using dual-ranked DIMMs often provides for better utilization of the memory channel and resulting in better bandwidth. If number of DRAM channels is increased, the workload can then be distributed between them for an added increase in bandwidth as well. CPU Prefetching allows the CPU to begin pulling in adjacent data into local cache while it is processing previous data, effectively reducing the access time to the total data. These factors influence the results of measuring latency and bandwidth.



Conclusion

In conclusion, using the utilities provided by the LMBench suite allows for proper measuring of latency and bandwidth across cache and system memory. This sound methodology provided by the utilities 'lat_mem_rd' and 'bw_mem' measure cache and memory latency and bandwidth. Knowing these values provide for a better understanding of the system performance. This becomes useful for comparing setups between different teams or verifying a customer setup. By making these values consistent between two separate platforms, additional benchmarking can be done with comfort in knowing these major factors of performance do not play a role when comparing. By converting latency measurements into CPU clocks, the speed of the processor no longer plays a factor in system memory accesses, thus making this number comparable to other system configurations. Because the utilities used are open source, and use the Linux operating system, there is no added cost to taking these measurements. The open source model also provides for the ability to recreate the benchmarks on a customer's site, between two groups in the same organization, or even within the same team on two separate platforms. There is no need for proprietary tools and Non Disclosure Agreements to get in the way of an issue or troubleshooting a problem.



Author

Joshua Ruggiero is a Computer Systems Engineer on the Performance Measurement and Analysis Team with the Embedded and Communications Group at Intel Corporation. He has a Bachelors of Science in Engineering from Arizona State University in Computer Systems Engineering.

Acronyms

CPU	Central Processing Unit
FSB	Front Side Bus
GB	Gigabyte (2^{30} bytes)
GHz	Gigahertz
KB	Kilobyte (2^{10} bytes)
L1	Level 1 Cache
L2	Level 2 Cache
MB	Megabyte (2^{20} bytes)
MHz	Megahertz
ns	nanoseconds
SoC	System on Chip



INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. Intel products are not intended for use in medical, life saving, or life sustaining applications.

Intel may make changes to specifications and product descriptions at any time, without notice.

This paper is for informational purposes only. THIS DOCUMENT IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE. Intel disclaims all liability, including liability for infringement of any proprietary rights, relating to use of information in this specification. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted herein.

Intel, the Intel logo, Intel leap ahead, Intel leap ahead logo, Quad-Core Intel Xeon Processor E5440, Intel 5100 Chipset, Xeon, EP80579, and Intel Core 2 Duo, are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2008 Intel Corporation. All rights reserved.

§