Intel® Xeon Phi™ Coprocessor - the Architecture

The first Intel® Many Integrated Core (Intel® MIC) architecture product

George Chrysos, Intel Corporation

This whitepaper is a transcription of George Chrysos' presentation at the Hot Chips conference held in September 2012, covering details about the Intel® Xeon Phi™ coprocessor, more specifically the first generation product (codenamed Knights Corner). Note that the results quoted in this paper were measured in development labs at Intel Corporation on prototype hardware and systems.

Intel® Many Integrated Core (Intel® MIC) architecture combines many Intel CPU cores onto a single chip. Intel MIC architecture is targeted for highly parallel, High Performance Computing (HPC) workloads in a variety of fields such as computational physics, chemistry, biology, and financial services. Today such workloads are run as task parallel programs on large compute clusters.

The Intel MIC architecture is aimed at achieving high throughput performance in cluster environments where there are rigid floor planning and power constraints. A key attribute of the microarchitecture is that it is built to provide a general-purpose programming environment similar to the Intel® Xeon® processor programming environment. The Intel Xeon Phi coprocessors based on the Intel MIC architecture run a full service Linux* operating system, support x86 memory order model and IEEE 754 floating-point arithmetic, and are capable of running applications written in industry-standard programming languages such as Fortran, C, and C++. The coprocessor is supported by a rich development environment that includes compilers, numerous libraries such as threading libraries and high performance math libraries, performance characterizing and tuning tools, and debuggers.

The Intel Xeon Phi coprocessor is connected to an Intel Xeon processor, also known as the "host", through a PCI Express (PCIe) bus. Since the Intel Xeon Phi coprocessor runs a Linux operating system, a virtualized TCP/IP stack could be implemented over the PCIe bus, allowing the user to access the coprocessor as a network node. Thus, any user can connect to the coprocessor through a secure shell and directly run individual jobs or submit batchjobs to it. The coprocessor also supports heterogeneous applications wherein a part of the application executes on the host while a part executes on the coprocessor.

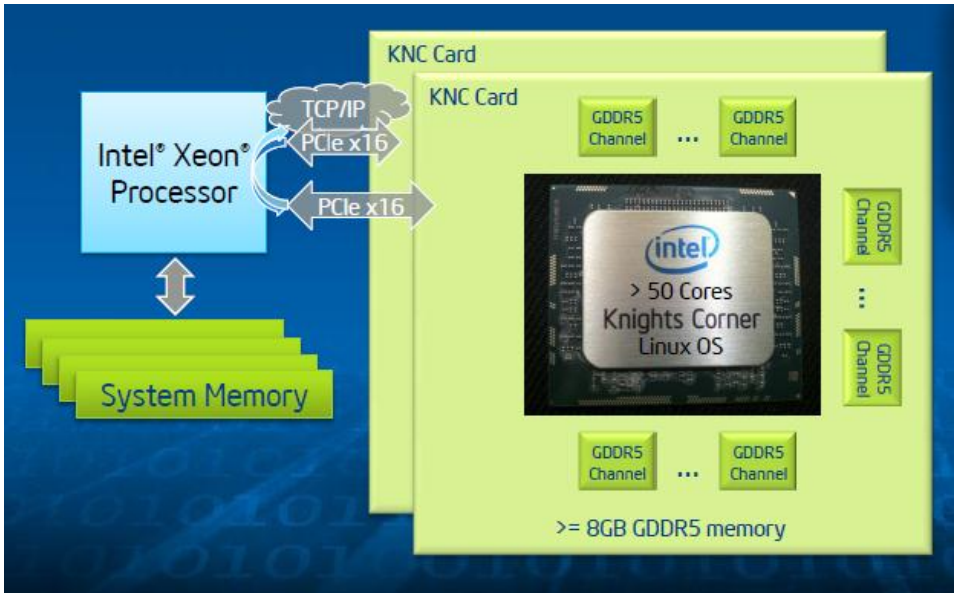


Figure 1. **The first generation Intel® Xeon Phi™ product codenamed "Knights Corner"**

Multiple Intel Xeon Phi coprocessors can be installed in a single host system. Within a single system, the coprocessors can communicate with each other through the PCIe peer-to-peer interconnect without any intervention from the host. Similarly, the coprocessors can also communicate through a network card such as InfiniBand or Ethernet, without any intervention from the host.

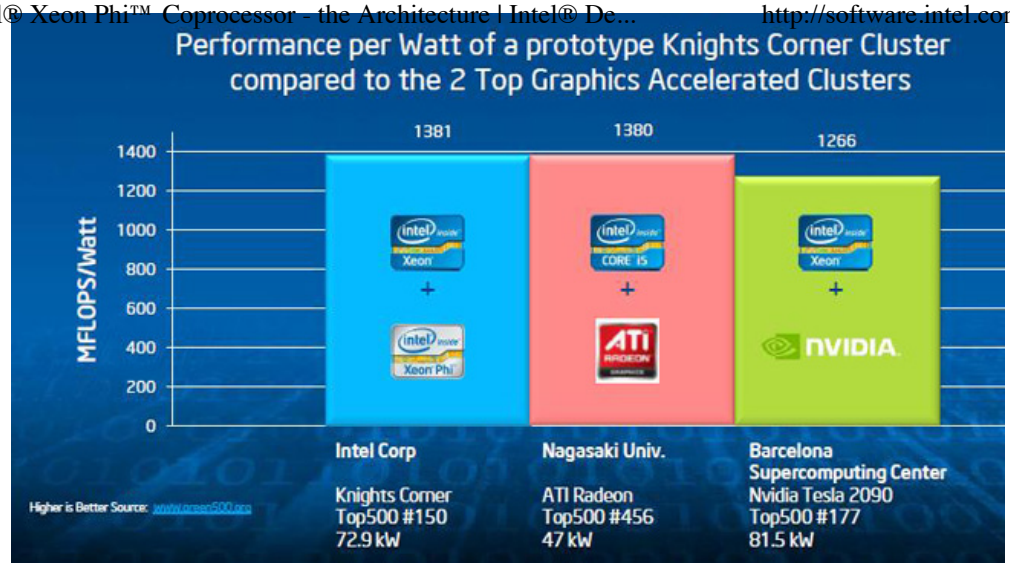
Note: Intel's released product actually contains over 60 cores (which is an update from the above graphic)

Intel's initial development cluster named "Endeavor", which is composed of 140 nodes of prototype Intel Xeon Phi coprocessors, was ranked 150 in the TOP500 supercomputers in the world based on its Linpack scores. Based on its power consumption (Figure 2), this cluster was as good if not better than other heterogeneous systems in the TOP500.

Figure 2. Linpack performance and power of Intel's cluster

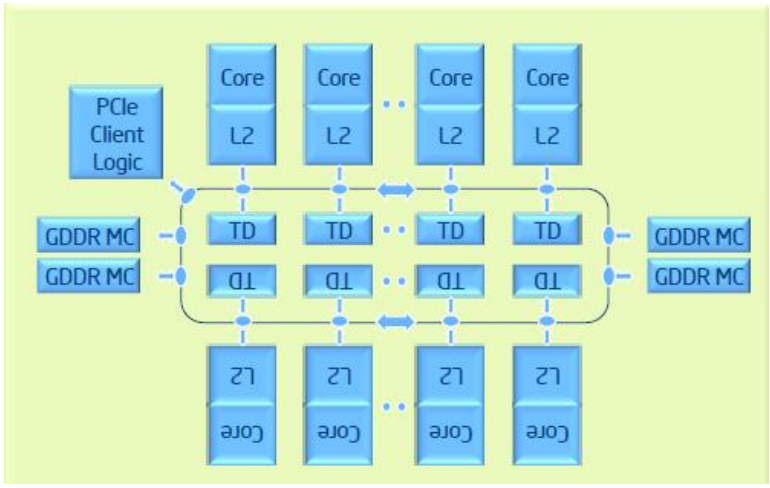
These results on unoptimized prototype systems demonstrate that high levels of performance efficiency can be achieved on compute-dense workloads without the need for a new programming language or APIs.

The Intel Xeon Phi coprocessor is primarily composed of processing cores, caches, memory controllers, PCIe client logic, and a very high bandwidth, bidirectional ring interconnect (Figure 3). Each core comes complete with a private L2 cache that is kept fully coherent by a



global-distributed tag directory. The memory controllers and the PCIe client logic provide a direct interface to the GDDR5 memory on the coprocessor and the PCIe bus, respectively. All these components are connected together by the ring interconnect.

Figure 3. Microarchitecture



Each core (Figure 4) in the Intel Xeon Phi coprocessor is designed to be power efficient while providing a high throughput for highly parallel workloads. A closer look reveals that the core uses a short in-order pipeline and is capable of supporting 4 threads in hardware. It is estimated that the cost to support IA architecture legacy is a mere 2% of the area costs of the core and is even less at a full chip or product level. Thus the cost of bringing the Intel Architecture legacy capability to the market is very marginal

Figure 4. Intel® Xeon Phi™ Coprocessor Core

Vector Processing Unit

An important component of the Intel Xeon Phi coprocessor's core is its vector processing unit (VPU), shown in Figure 5. The VPU features a novel 512-bit SIMD instruction set, officially known as Intel® Initial Many Core Instructions (Intel® IMCI). Thus, the VPU can execute 16 single-precision (SP) or 8 double-precision (DP) operations per cycle. The VPU also supports Fused Multiply-Add (FMA) instructions and hence can execute 32 SP or 16 DP floating point operations per cycle. It also provides support for integers.

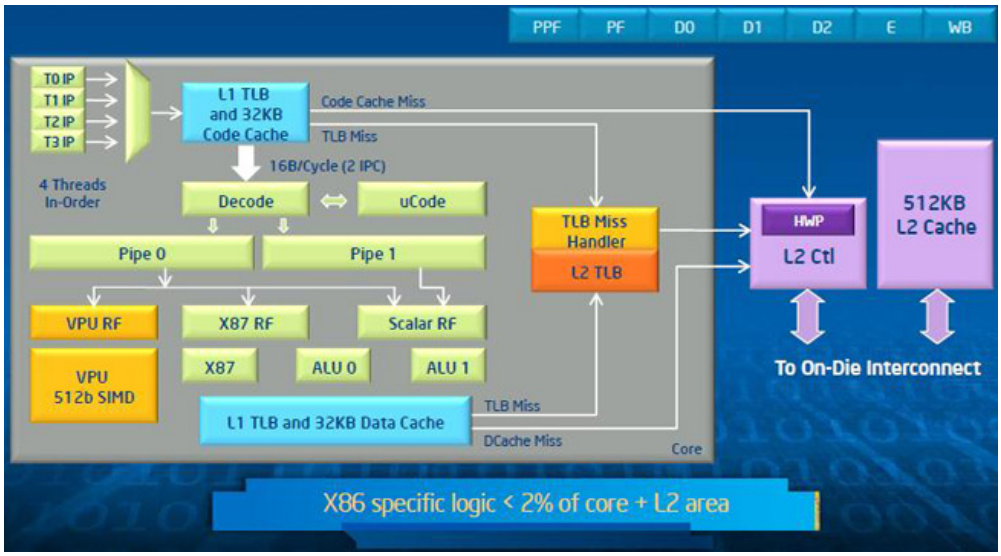
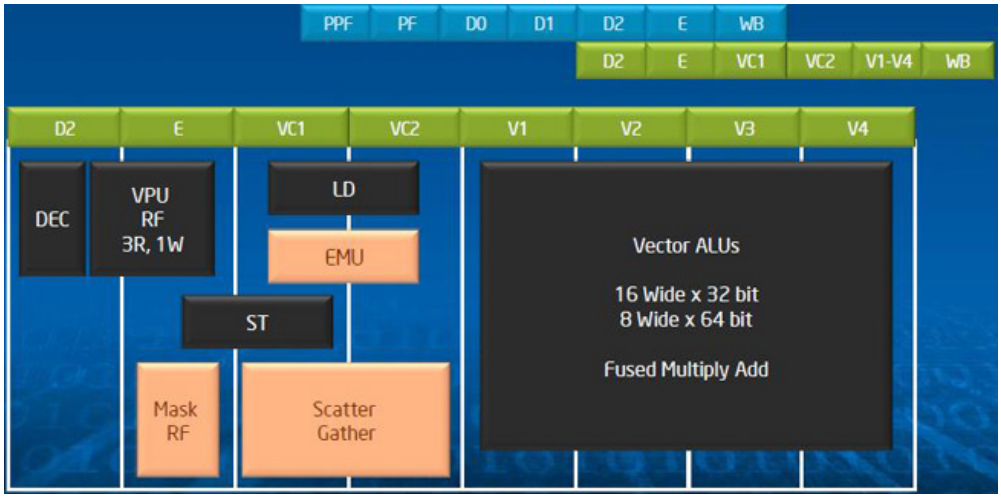


Figure 5. Vector Processing Unit



Vector units are very power efficient for HPC workloads. A single operation can encode a great deal of work and does not incur energy costs associated with fetching, decoding, and retiring many instructions. However, several improvements were required to support such wide SIMD instructions. For example, a mask register was added to the VPU to allow per lane predicated execution. This helped in vectorizing short conditional branches, thereby improving the overall software pipelining efficiency. The VPU also supports

gather and scatter instructions, which are simply non-unit stride vector memory accesses, directly in hardware. Thus for codes with sporadic or irregular access patterns, vector scatter and gather instructions help in keeping the code vectorized.

The VPU also features an Extended Math Unit (EMU) that can execute transcendental operations such as reciprocal, square root, and log, thereby allowing these operations to be executed in a vector fashion with high bandwidth. The EMU operates by calculating polynomial approximations of these functions.

The Interconnect

The interconnect (Figure 6) is implemented as a bidirectional ring. Each direction is comprised of three independent rings. The first, largest, and most expensive of these is the data block ring. The data block ring is 64 bytes wide to support the high bandwidth requirement due to the large number of cores. The address ring is much smaller and is used to send read/write commands and memory addresses. Finally, the smallest ring and the least expensive ring is the acknowledgement ring, which sends flow control and coherence messages.

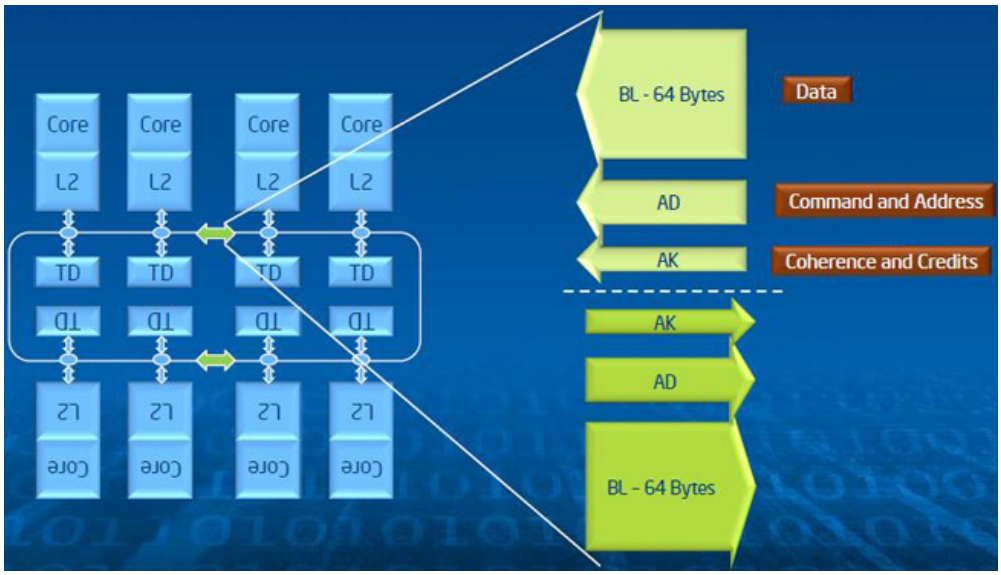


Figure 6. The Interconnect

When a core accesses its L2 cache (Figure 7) and misses, an address request is sent on the address ring to the tag directories. The memory addresses are uniformly distributed amongst the tag directories on the ring to provide a smooth traffic characteristic on the ring. If the requested data block is found in another core's L2 cache, a forwarding request is sent to that core's L2 over the address ring and the request block is subsequently forwarded on the data block ring. If the requested data is not found in any caches, a memory address is sent from the tag directory to the memory controller.

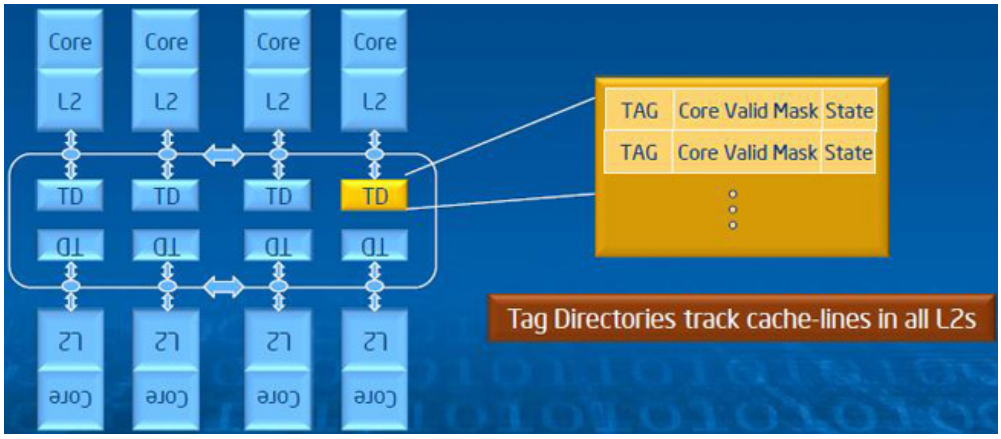


Figure 7. Distributed Tag Directories

Figure 8 shows the distribution of the memory controllers on the bidirectional ring. The memory controllers are symmetrically interleaved around the ring. . There is an all-to-all mapping from the tag directories to the memory controllers. The addresses are evenly distributed across the memory controllers, thereby eliminating hotspots and providing a uniform access pattern which is essential for a good bandwidth response.

Figure 8. Interleaved Memory Access

During a memory access, whenever an L2 cache miss occurs on a core, the core generates an address request on the address ring and queries the tag directories. If the data is not found in the tag directories, the core generates another address request and queries the memory for the data. Once the memory controller fetches the data block from memory, it is returned back to the core over the data ring. Thus during this process, one data block, two address requests (and by protocol, two acknowledgement messages) are transmitted over the rings. Since the data block rings are the most expensive and are designed to support the required data bandwidth, we need to increase the number of less expensive address and acknowledgement rings by a factor of two to match the increased bandwidth requirement caused by the higher number of requests on these rings (Figure 9).

Figure 9. Interconnect: 2x AD/AK

Multi-Threaded Streams Triad

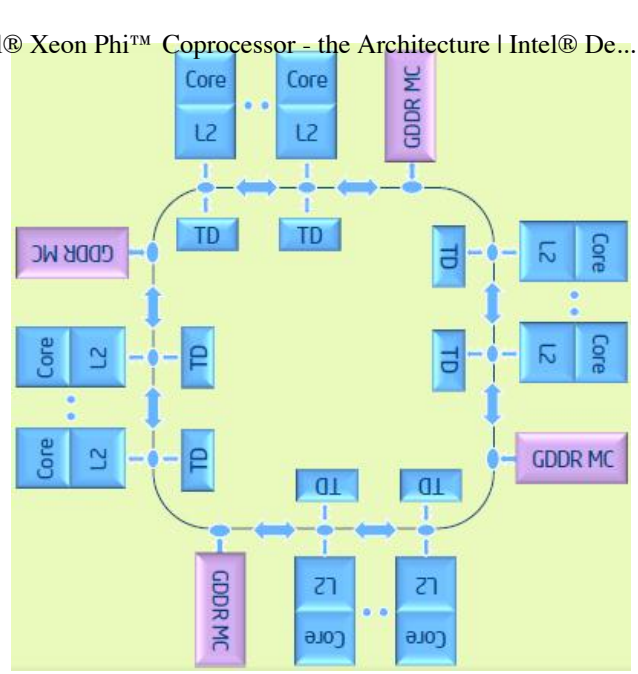
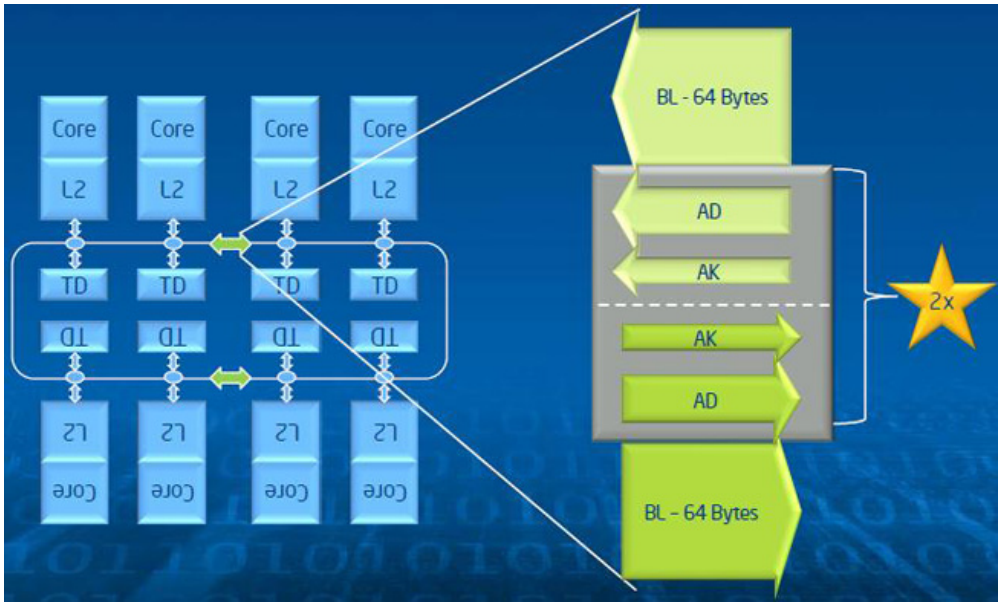


Figure 10 shows the core scaling results for the multi-threaded streams triad workload. These results were generated on a simulator for a prototype of the Intel Xeon Phi coprocessor with only one address ring and one acknowledgement ring per direction in its interconnect. The results indicate that in this case the address and acknowledgement rings would become performance bottlenecks and would exhibit poor scalability beyond 32 cores.

Figure 10. Multi-threaded Triad – Saturation for 1 AD/AK Ring

The production grade Intel Xeon Phi coprocessor uses two address and two acknowledgement rings per direction and provides a good performance scaling up to 50 cores and beyond, as shown in Figure 11. It is evident from the figure that the addition of the rings results in an over 40% aggregate bandwidth improvement.

Figure 11. Multi-threaded Triad – Benefit of Doubling AD/AK



Streaming Stores

Streaming stores was another key innovation that was employed to further boost memory bandwidth. The pseudo code for Streams Triads is shown below:

```
Streams Triad
  for (I=0; I<HUGE; I++)
    A[I] = k*B[I] + C[I];
```

The stream triad kernel reads two arrays, B and C, and writes a single array A from memory. Historically, a core reads a cache line before it writes the addressed data. Hence there is an additional read overhead associated with the write. A streaming store instruction allows the cores to write an entire cache line without reading it first. This reduces the number of bytes transferred per iteration from 256 bytes to 192 bytes (Table 1).

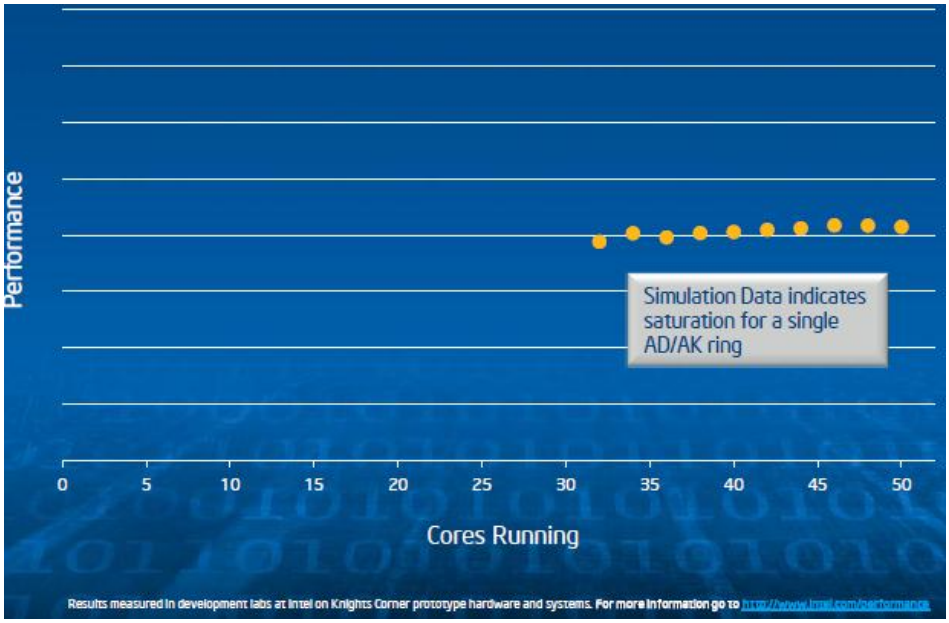


Table 1. Streaming Stores

	Without Streaming Stores	With Streaming Stores
Behavior	Read A, B, C, write A	Read B, C, write A
Bytes transferred to/from memory per iteration	256	192

Figure 12 shows the core scaling results of stream triads workload with streaming stores. As is evident from the results, streaming stores provide a 30% improvement over previous results. Totally, then, by adding two rings per direction and implementing streaming stores we are able to improve bandwidth by more than a factor of 2 for multithreaded streams triad.

Figure 12. Multi-threaded Triad – with Streaming Stores

Other Design Features

Other micro-architectural optimizations incorporated into the Intel Xeon Phi coprocessor include a 64-entry second-level Translation Lookaside Buffer (TLB), simultaneous data cache loads and stores, and 512KB L2 caches. Lastly, the Intel Xeon Phi coprocessor implements a 16 stream hardware prefetcher to improve the cache hits and provide higher bandwidth. Figure 13 shows the net performance improvements for the SPECfp 2006 benchmark suite for a single core, single thread runs. The results indicate an average improvement of over 80% per cycle not including frequency.

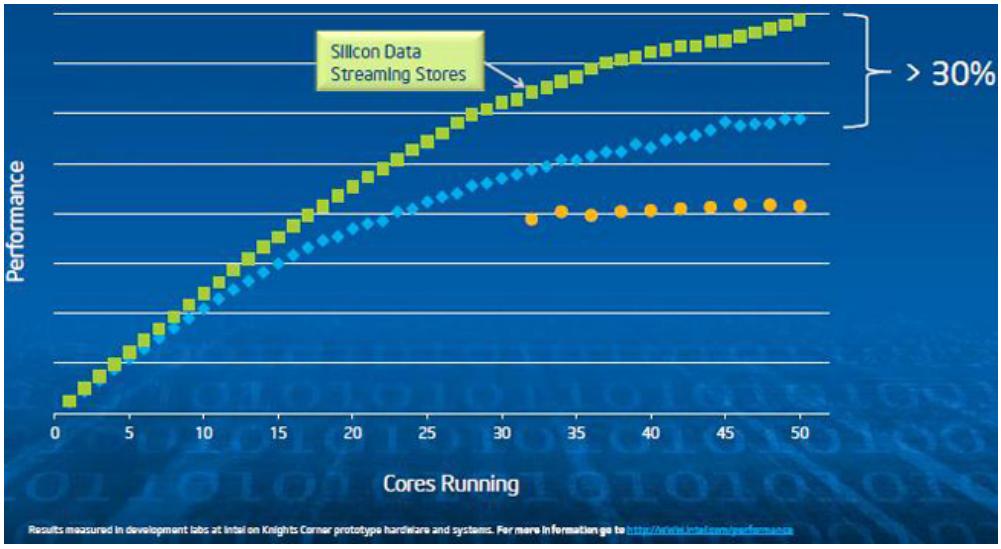


Figure 13. Per-Core ST Performance Improvement (per cycle)

Caches

The Intel MIC architecture invests more heavily in L1 and L2 caches compared to GPU architectures. The Intel Xeon Phi coprocessor implements a leading-edge, very high bandwidth memory subsystem. Each core is equipped with a 32KB L1 instruction cache and 32KB L1 data cache and a 512KB unified L2 cache. These caches are fully coherent and implement the x86 memory order model. The L1 and L2 caches provide an aggregate bandwidth that is approximately 15 and 7 times, respectively, faster compared to the aggregate memory bandwidth. Hence, effective utilization of the caches is key to achieving peak performance on the Intel Xeon Phi coprocessor. In addition to improving bandwidth, the caches are also more energy efficient for supplying data to the cores than memory. . Figure 14 shows the energy consumed per byte of data transfered from the memory, and L1 and L2 caches. In the exascale compute era, caches will play a crucial role in achieving real performance under restrictive power constraints.

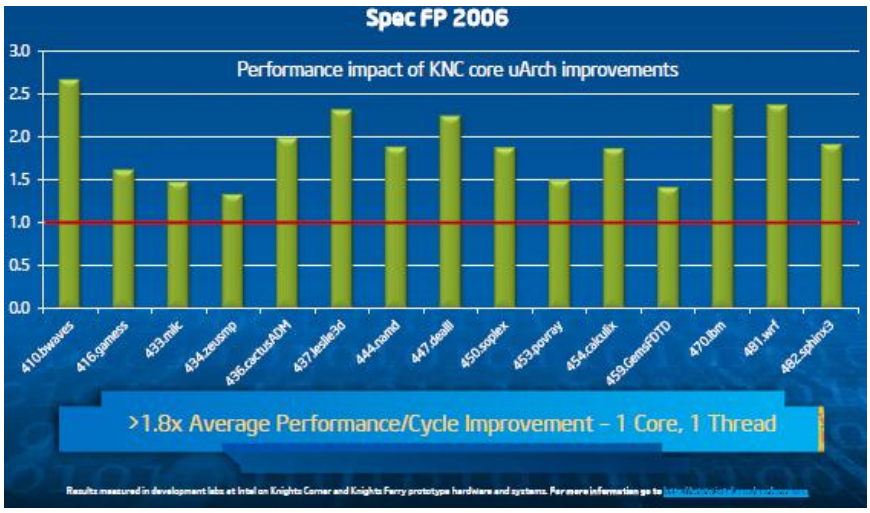


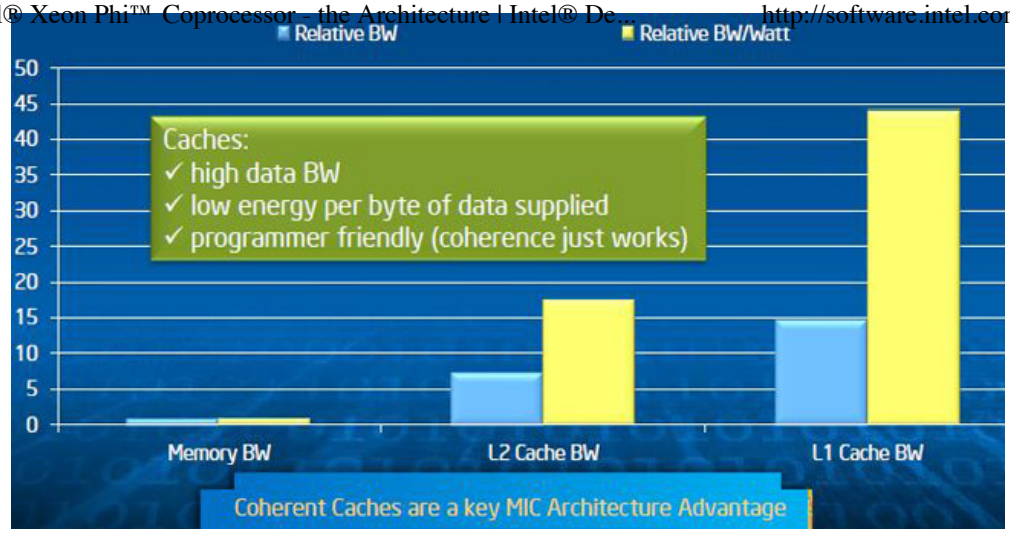
Figure 14. Caches – For or Against?

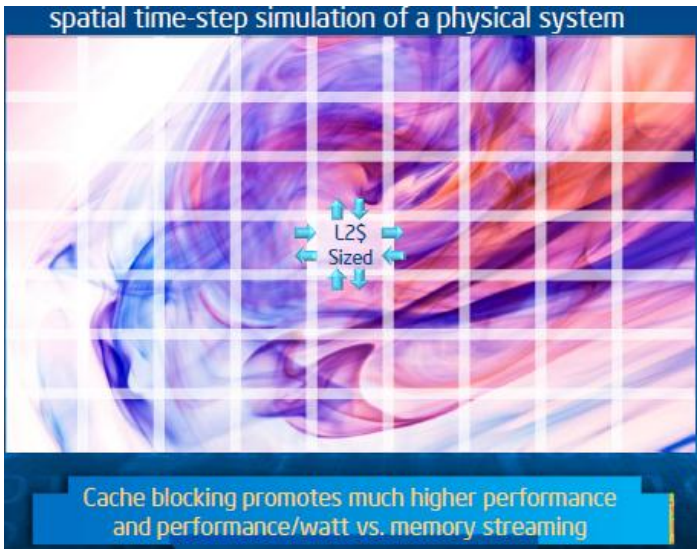
Stencils

Stencils (Figure 15) are common in physics simulations and are classic examples of workloads which show a large performance gain through efficient use of caches.

Figure 15. Stencils Example

Stencils are typically employed in simulation of physical systems to study the behavior of the system over time. When these workloads are not





programmed to be cache-blocked, they will be bound by memory bandwidth. Cache blocking promises substantial performance gains given the increased bandwidth and energy efficiency of the caches compared to memory. Cache blocking improves performance by blocking the physical structure or the physical system such that the blocked data fits well into a core's L1 and or L2 caches. For example, during each time-step, the same core can process the data which is already resident in the L2 cache from the last time step, and hence does not need to be fetched from the memory, thereby improving performance. Additionally, the cache coherence further aids the stencil operation by automatically fetching the updated data from the nearest neighboring blocks which are resident in the L2 caches of other cores. Thus, stencils clearly demonstrate the benefits of efficient cache utilization and coherence in HPC workloads.

Power Management

Figure 16. Power Management: All On and Running

Intel Xeon Phi coprocessors are not suitable for all workloads. In some cases, it is beneficial to run the workloads only on the host. In such situations where the coprocessor is not being used, it is necessary to put the coprocessor in a power-saving mode. Figure 16 shows all the components of the Intel Xeon Phi coprocessor in a running state. To conserve power, as soon as all the four threads on a core are halted, the clock to the core is gated (Figure 17). Once the clock has been gated for some programmable time, the core power gates itself, as shown in Figure 18, thereby eliminating any leakage.

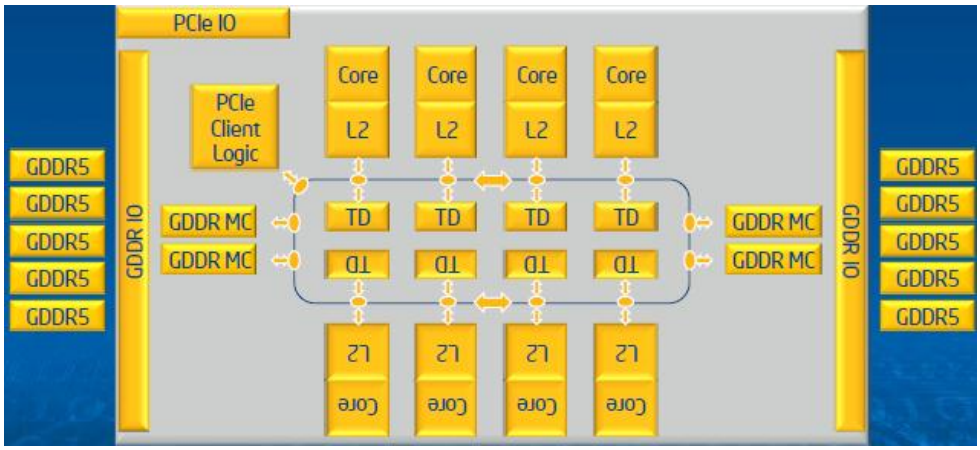


Figure 17. Core C1: Clock Gate Core

At any point, any number of the cores can be powered down or powered up as shown in Figure 18. Additionally, when all the cores are power gated and the uncore detects no activity, the tag directories, the interconnect, L2 caches and the memory controllers are clock gated.

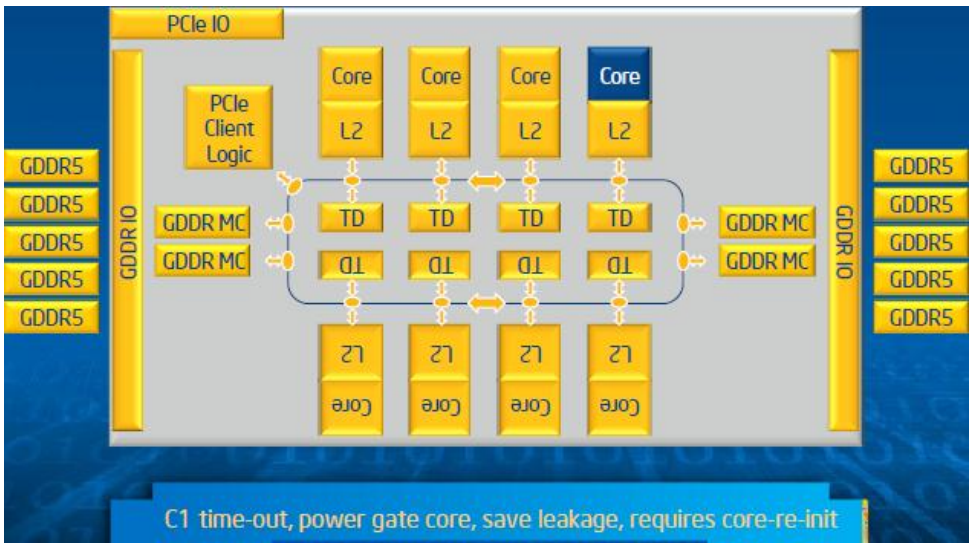
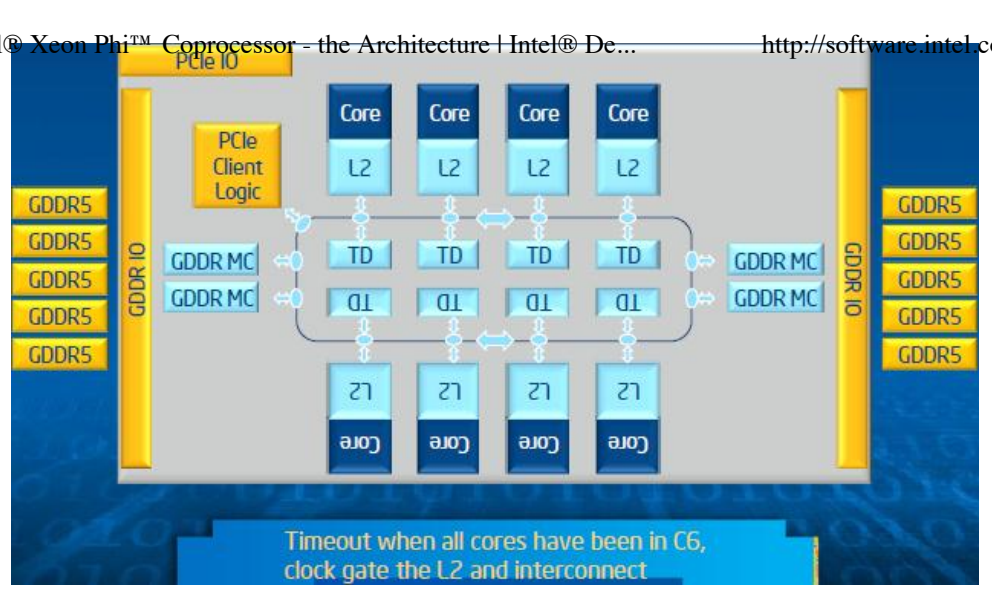
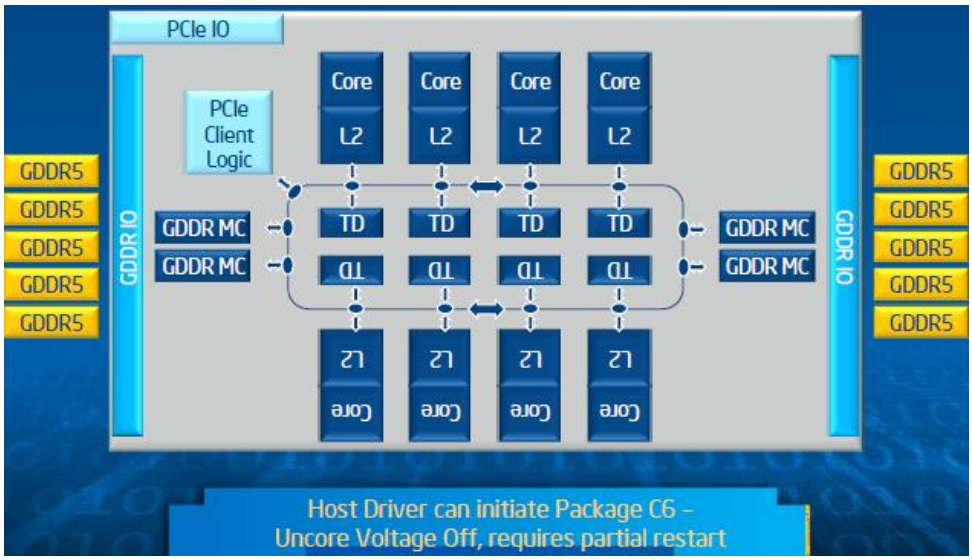


Figure 18. Core C6: Power Gate Core

At this point, the host driver can put the coprocessor into a deeper sleep or an idle state, wherein all the uncore is power gated, the GDDR is put into a self-refresh mode, the PCIe logic is put in a wait state for a





wakeup and the GDDR-IO is consuming very little power (Figure 19). These power management techniques help conserve power and make Intel Xeon Phi coprocessor an excellent candidate for data centers.

Figure 19. Package Auto C3

Summary

The Intel Xeon Phi coprocessor provides high performance, and performance per watt for highly parallel HPC workloads, while not requiring a new programming model, API, language or restrictive memory modelIt is able to do this with an array of general purpose cores withmultiple thread contexts, wide vector units, caches, and high bandwidth on die and memory interconnect. Knights Corner is the first Intel Xeon Phi product in the MIC architecture family of processors from Intel, aimed at enabling the exascale era of computing.

Notices

Performance Notice

For more complete information about performance and benchmark results, visit **www.intel.com/benchmarks (http://www.intel.com/benchmarks)**

Categories: **Intel® Many Integrated Core Architecture , Developers , Linux* , Intermediate**

For more complete information about compiler optimizations, see our **Optimization Notice**.

RSS