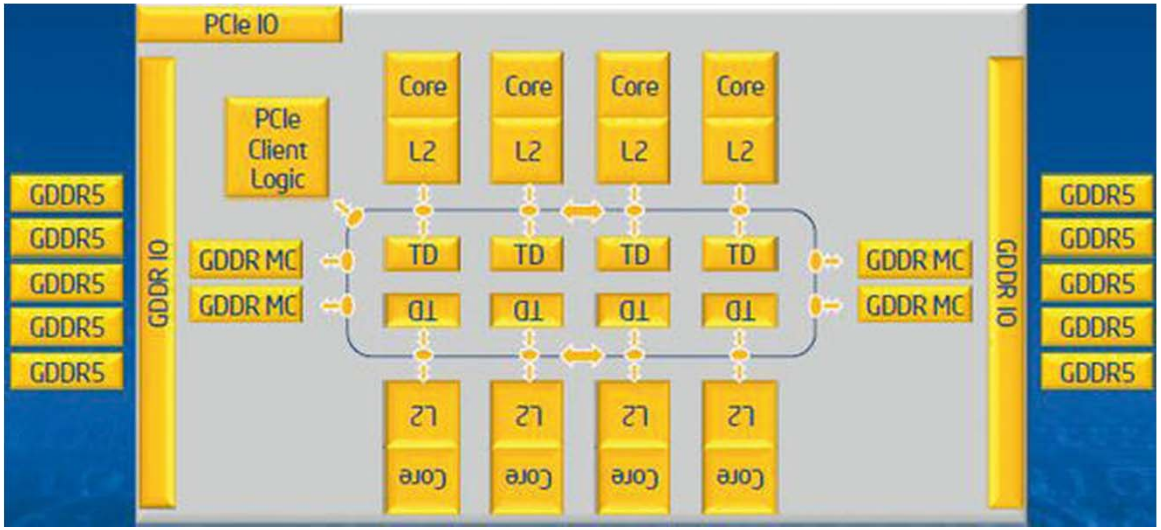
Lecture 18 – Accelerators: Intel Xeon Phi

- **The Intel Mic (Xeon Phi) is a shared-memory vector co-processor**
- **Uses a multi-core paradigm**
- **Cores are light-weight compared to usual cores on CPU**
- **Many light-weight cores (~61 with 244 threads) rather than few more-powerful cores on CPU (~16)**

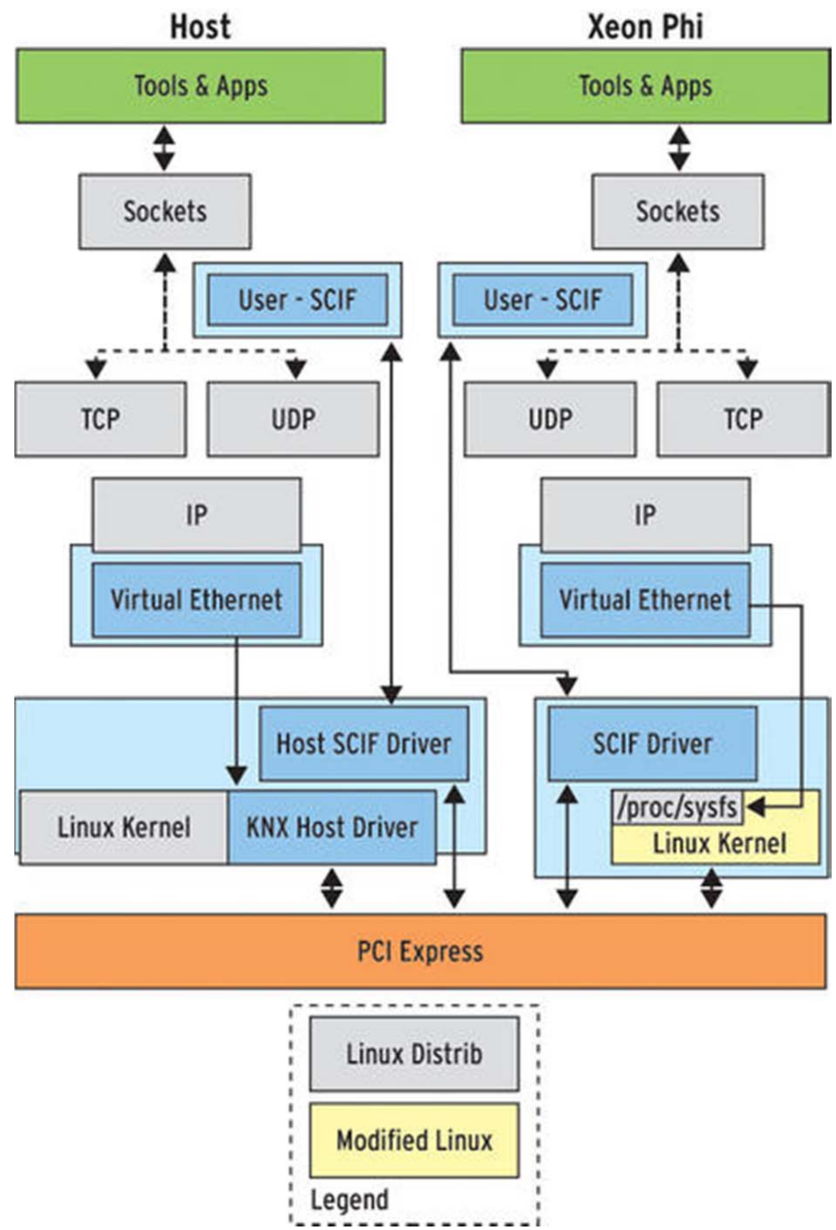
Architecture

- PCI Express card
- CPU cores based on Pentium technology
- 64-bit, floating-point instructions, vector unit with 32 512-bit registers
- Each core is multi-threaded four times
- 7100 series with 61 cores can run up to 244 threads at a time



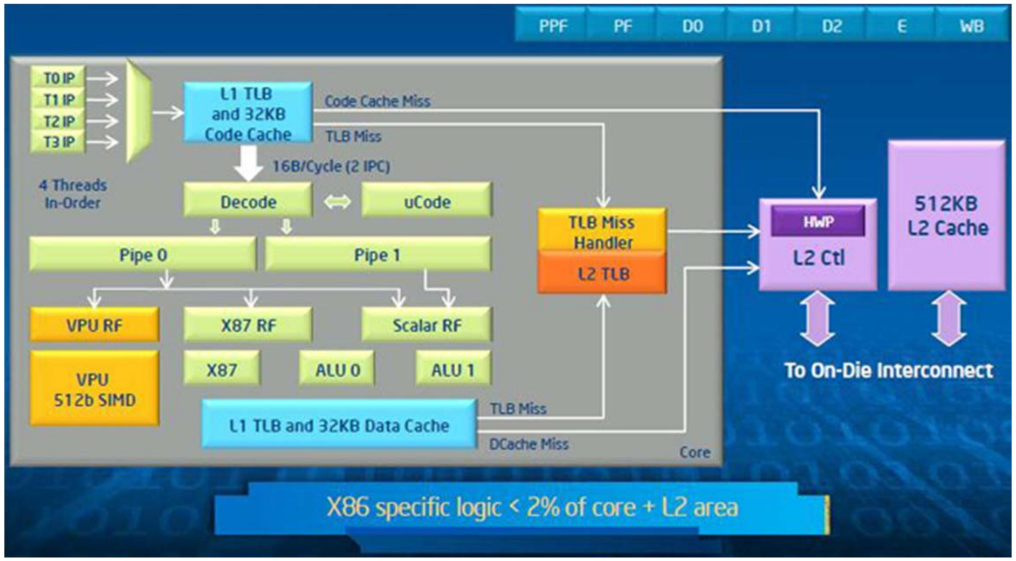
Architecture

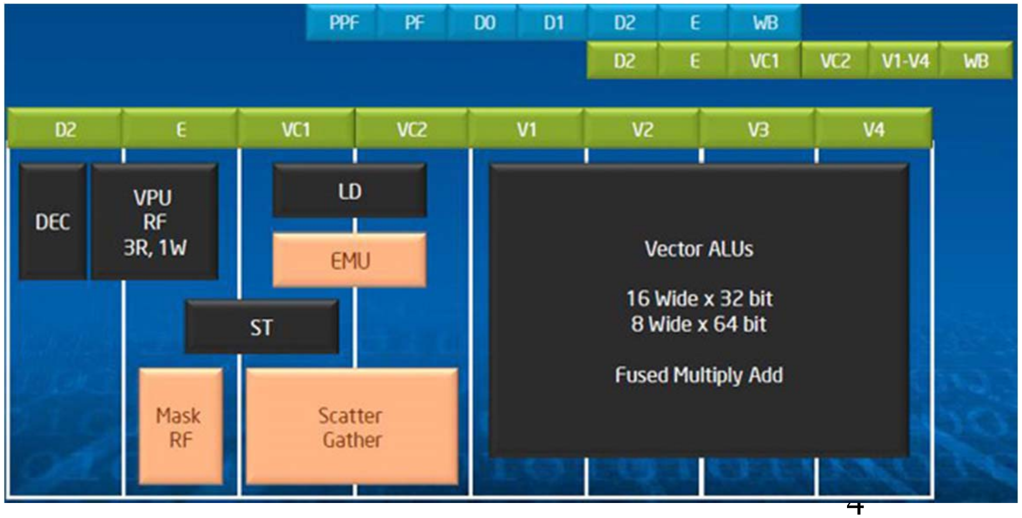
- **X86 Cores have 64 KB L1 cache and 512 KB L2 cache**
- **Interconnect is a ring bus**
- **Has it's own memory system**
- **Communication to/from host through a PCIe bus**
  - Symmetric Communications Interface (SCIF)
  - MPI can also be used (Phi looks like another processor to system)



Phi Cores

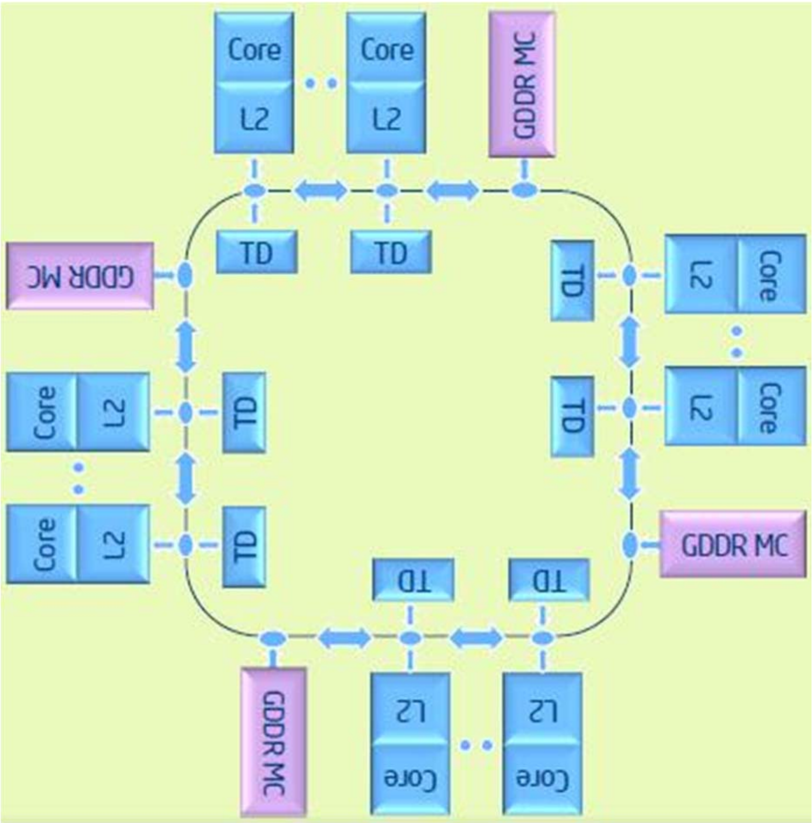
- **Core uses a short in-order pipeline that supports 4 threads in hardware**
- **L1 cache is part of core**
- **Vector Processing Unit (VPU) is part of core**
  - 512-bit instruction set
  - 16 single-precision or 8 double-precision operations per cycle





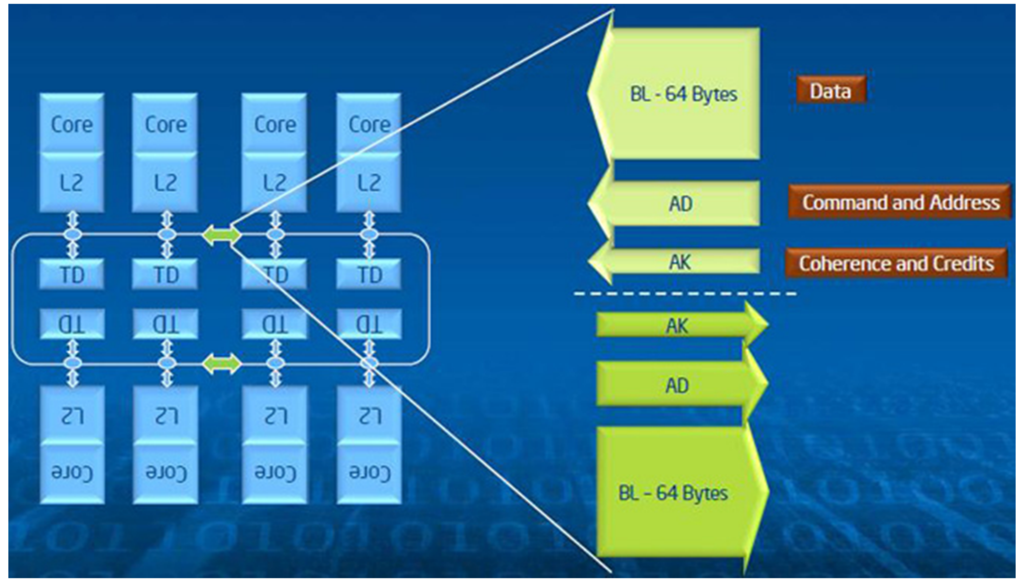
Phi Memory

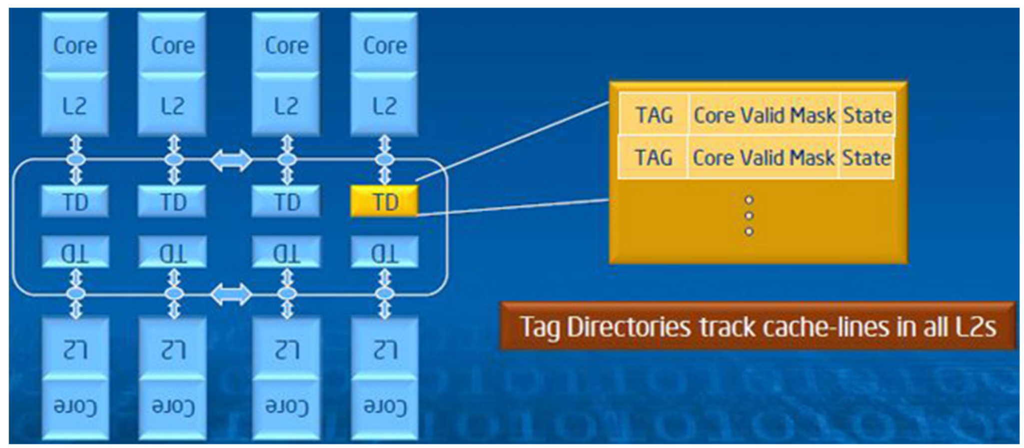
- Like GPU, Xeon Phi has low memory
- 5100 series has 35 MB of RAM for each thread
- Memory controllers use all-to-all mapping evenly distributed



Phi Interconnect

- **Bidirectional ring consisting of 3 independent rings (SCIF – Symmetric Communication InterFace)**
  - Data block ring (support high bandwidth)
  - Address ring (send read/write commands)
  - Acknowledgement ring (flow control and coherence messages)
- **Tag Directories (TD) used to track L2 cache-lines**





Programming on Phi

- **Intel MPI is used to program parallel codes on the Phi**
  - Based on ANL MPIC2 routines
- **3 Programming Modes**
  - Co-processor only – MPI ranks reside solely on coprocessor
  - Symmetric – MPI ranks reside on the host and the coprocessors
  - MPI Offload – MPI ranks reside solely on the host

Programming on Phi

- **2 Application Programming Models supported**
  - User-controlled
    - User defined sections of code loops or
    - SPMD routine(s)
    - Using OpenMP in combination with MPI when running on multiple co-processors
  - Automated break-up of loops using OpenMP
- **The Intel-Phi co-processor can use**
  - Shared memory toolsets (OpenMP) for the co-processors on the same Phi
  - Combination of shared memory (OpenMP) with distributed memory toolsets (Intel-MPI) for the co-processors on different Phis

Example – Calculation of Pi

Courtesy of Loc Q Nguyen (Intel)

- The sample program estimates the calculation of Pi (π) using a Monte Carlo method. The symmetric model is used giving ranks to both hosts and co-processors
- Consider a sphere centered at the origin and circumscribed by a cube: the sphere's radius is r and the cube edge length is 2r. The volumes of a sphere and a cube are given by

$$V_{sphere} = \frac{4\pi r^3}{3}$$

$$V_{cube} = (2r)^3 = 8r^3$$

Example – Calculation of Pi

Courtesy of Loc Q Nguyen (Intel)

- The first octant of the coordinate system contains one eighth of the volumes of both the sphere and the cube; the volumes in that octant are given by:

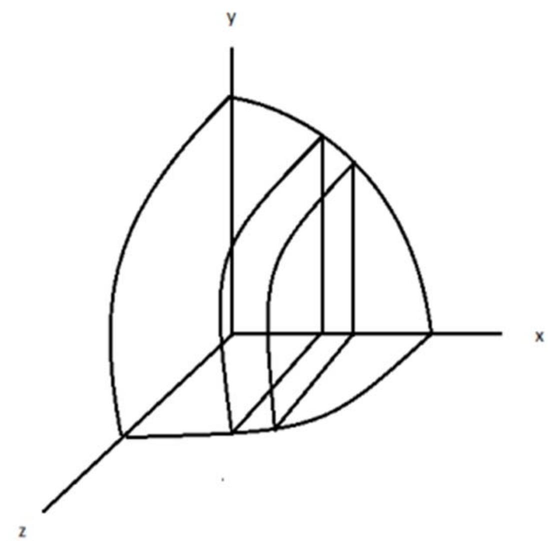
$$V_{sphere} = \frac{\pi r^3}{6}$$

$$V_{cube} = r^3$$

Example – Calculation of Pi
Courtesy of Loc Q Nguyen (Intel)

- If we generate *Nc* points uniformly and randomly in the cube within this octant, we expect that about *Ns* points will be inside the volume of sphere according to the following ratio:

$$\frac{N_c}{N_s} = \frac{6r^3}{\pi r^3} = \frac{6}{\pi}$$



- Therefore, the estimated Pi (π) is calculated by

$$\pi = \frac{6N_s}{N_c}$$

where N_c is the number of points generated in the portion of the cube residing in the first octant, and N_s is the total number of points found inside the portion of the sphere residing in the first octant.

Example – Calculation of Pi

Courtesy of Loc Q Nguyen (Intel)

- In the implementation, rank 0 (process) is responsible for dividing the work among the other *n* ranks. Each rank is assigned a chunk of work, and the summation is used to estimate the number Pi.
- Rank 0 divides the *x-axis* into *n* equal segments. Each rank generates (N_C/n) points in the assigned segment, and then computes the number of points in the first octant of the sphere.

Pseudo Code

Courtesy of Loc Q Nguyen (Intel)

Rank 0 generate n random seed

Rank 0 broadcast all random seeds to n rank

For each rank i [0, n-1]

    receive the corresponding seed

    set num_inside = 0

    For j=0 to Nc / n

        generate a point with coordinates

            x between [i/n, (i+1)/n]

            y between [0, 1]

            z between [0, 1]

        compute the distance d = x^2 + y^2 + z^2

        if distance d <= 1, increment num_inside

    Send num_inside back to rank 0

Rank 0 set Ns to the sum of all num_inside

Rank 0 compute Pi = 6 $*$ Ns / Nc

Code

- **See Appendix of "Using_Intel_MPI_on_Intel_Xeon_Phi_Coprocessor_Systems-v1_4" on Smartsite Additional Material/Intel_Xeon_Phi**

Compiling the C-Code

Courtesy of Loc Q Nguyen (Intel)

- mpiicc –mmic montecarlo.c -o montecarlo.mic
- mpiicc montecarlo.c -o montecarlo.host

- Enable the MPI communication between host and coprocessors:
- # export I_MPI_MIC=enable

- Running on a single host
- mpirun –n <*# of processes*> -host <*hostname*> <*application*>

- *Running on multiple hosts (or use a –machinefile)*
- mpirun –n <*# of processes*> -host <*hostname1*> <*application*> : –n <*# of processes*> -host <*hostname2*> <*application*>

Results of Code

```
mpirun -n 3 -host mic0 /tmp/montecarlo.mic : -n 5 -host mic1 \ /tmp/montecarlo.mic
Hello world: rank 0 of 8 running on knightscorner0-mic0
Hello world: rank 1 of 8 running on knightscorner0-mic0
Hello world: rank 2 of 8 running on knightscorner0-mic0
Hello world: rank 3 of 8 running on knightscorner0-mic1
Hello world: rank 4 of 8 running on knightscorner0-mic1
Hello world: rank 5 of 8 running on knightscorner0-mic1
Hello world: rank 6 of 8 running on knightscorner0-mic1
Hello world: rank 7 of 8 running on knightscorner0-mic1
Elapsed time from rank 0: 255.25 (sec)
Elapsed time from rank 1: 241.74 (sec)
Elapsed time from rank 2: 245.78 (sec)
Elapsed time from rank 3: 241.82 (sec)
Elapsed time from rank 4: 256.23 (sec)
Elapsed time from rank 5: 241.42 (sec)
Elapsed time from rank 6: 240.81 (sec)
Elapsed time from rank 7: 240.10 (sec)
Out of 4294967295 points, there are 2248825514 points inside the sphere => pi= 3.141572952271
```

Xeon Intel Phi – Advantages/Disadvantages

- **Advantages: Developers of Phi have worked with Intel compiler groups to integrate the Phi into MPI and OpenMP programming models**
  - Cores on Phi have ranks similar to those on hosts
  - Send and Recv MPI commands can be used in the same fashion as on multi-core hosts
  - OpenMP pragma directives also work in the same fashion as on multi-core hosts
- **Disadvantage: Developers are dependent on Intel's proprietary compiler, Intel-MPI**

Additional Information

- **There are several websites where additional information can be found including:**

https://software.intel.com/en-us/blogs/2013/05/01/code-examples-from-xeon-phi-book?language=it

https://www.msi.umn.edu/sites/default/files/Phi_Intro.pdf