Intel Xeon Phi Processor

Binyao Jiang, Michael Schoen, Jingyuan Zhang

Brief introduction of Xeon Phi[4]

- Origin

  First generation: PCIe coprocessor
  Second generation: a standard CPU

- Performance

  Good sequential and parallel performance

- Many-cores Processor

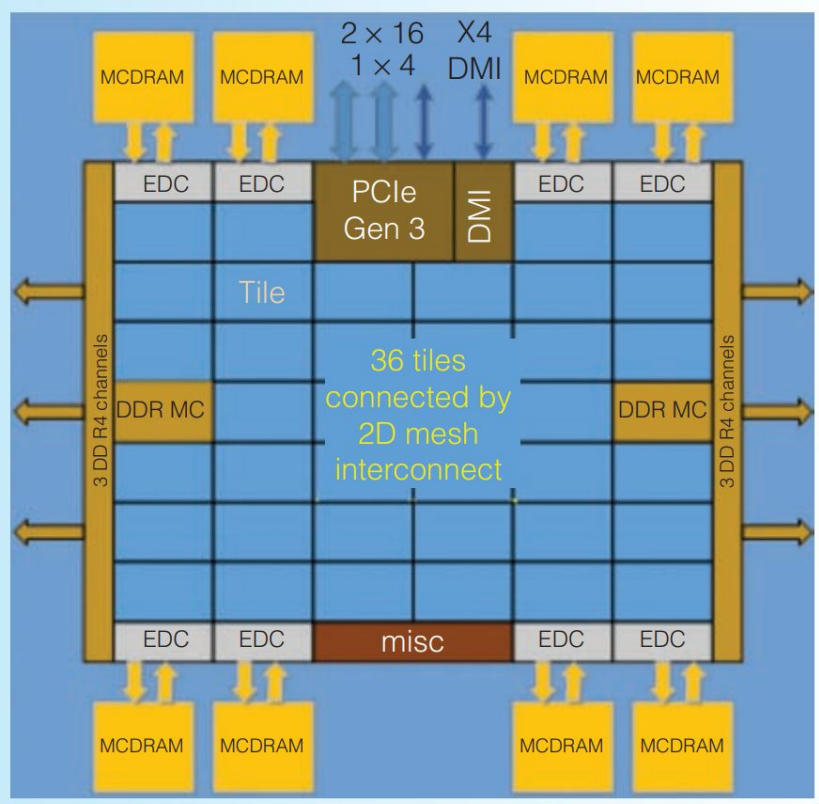
  The Xeon Phi has 72 cores. Compare that to an Intel-9900K (a modern high-end desktop processor) which has 8.

The 5 main components[1]

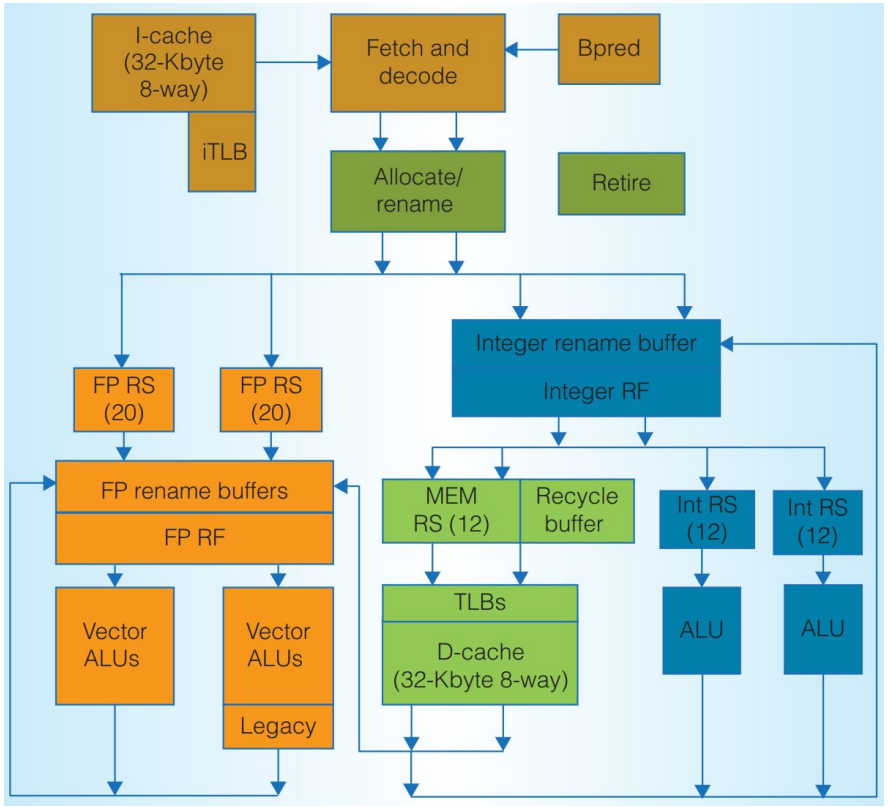
The Xeon Phi is made up 32 tiles. Each of these tiles is made up with 5 main components

1) Front-end unit
    a) Handles fetching and decoding instructions
2) Allocation unit
    a) Assigns necessary pipelines resources
3) Integer Execution Unit
    a) Executes integer micro-ops
4) Memory Execution Unit
    a) Executes Memory micro-ops, as well as instruction cache misses
5) Vector processing unit
    a) Executes floating-point and integer division instructions

Title Architecture



Core and VPU Dataflow Block Diagram



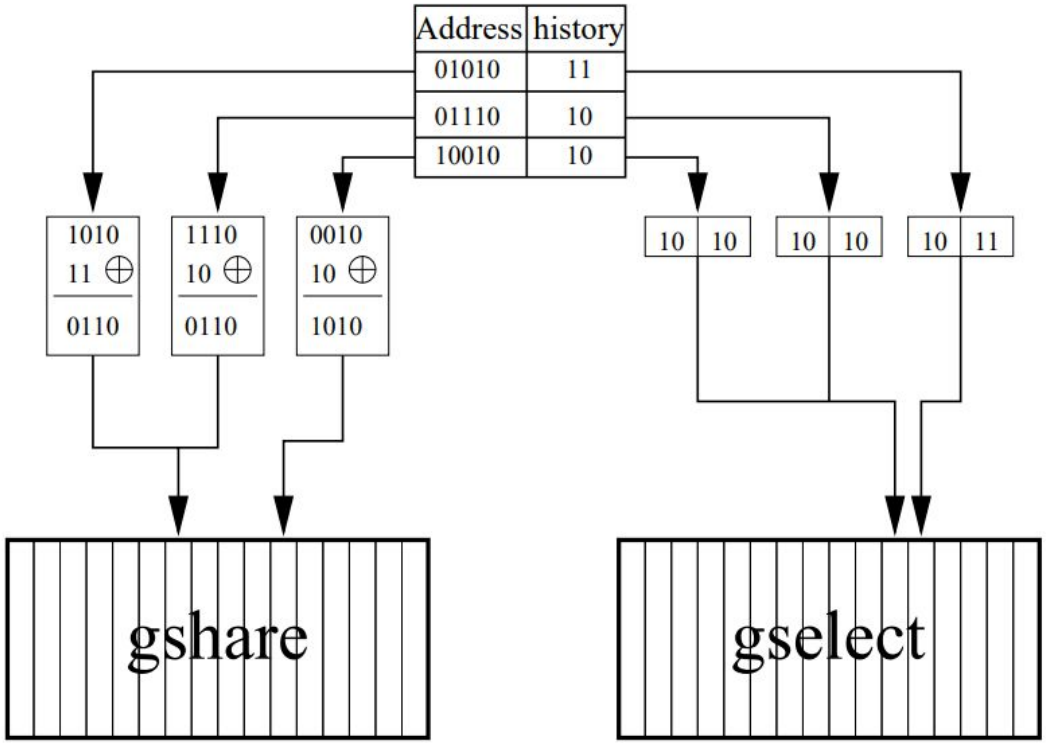
Front-end Unit Details

- 32-Kbyte instruction cache
- 48-entry instruction TLB
- Dual Issue
- Translates CISC instructions into RISC instructions (aka micro-ops)
- Uses a gskew-style branch predictor (more on that on the next slide)
- Micro-ops are placed into a 32-entry instruction queue for the allocation unit

Gskew Branch Prediction[2]

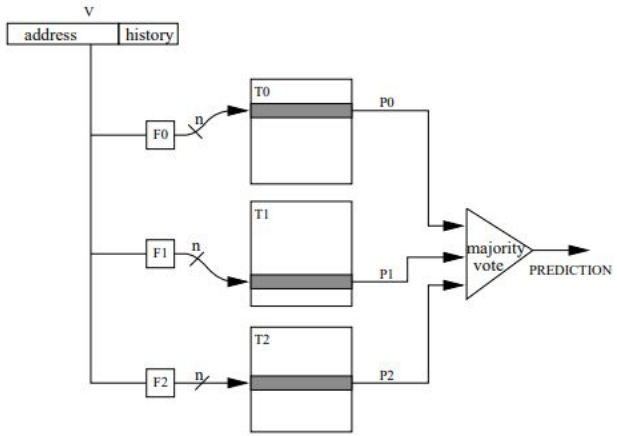
- As we learned in class, correlating predictors combine address and global information to provide better prediction
- Exactly how we combine the global history and address effects aliasing behavior
    - This method is known as the mapping function
- Two mapping function examples are gselect, which concatenates address and history, and gshare, which XOR’s address and history

Gskew Branch Prediction



Gskew Branch Prediction

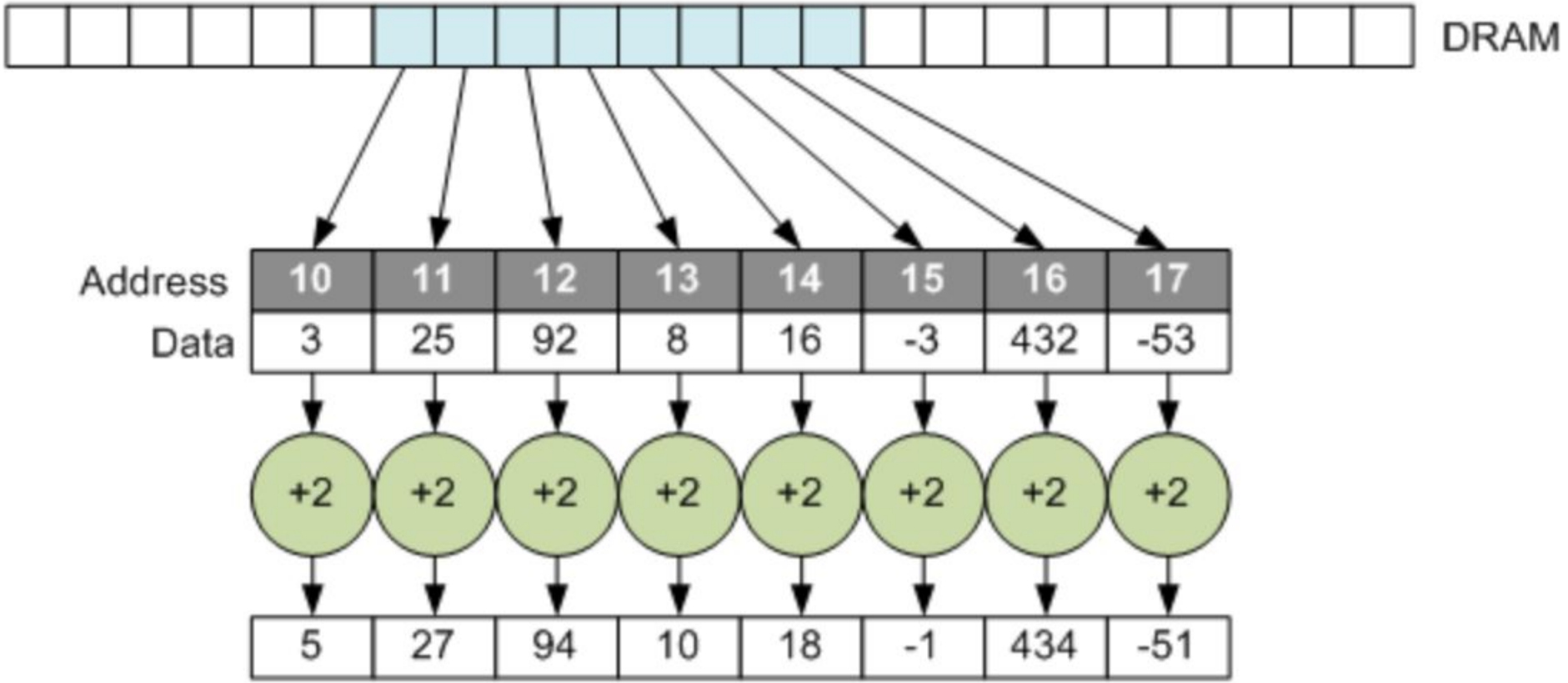
- Gskew takes advantage of fact that different mapping functions lead to different conflicts
- Gskew uses an odd number of branch-predictor banks, each with a different mapping function
- A majority vote is used to select a final branch direction



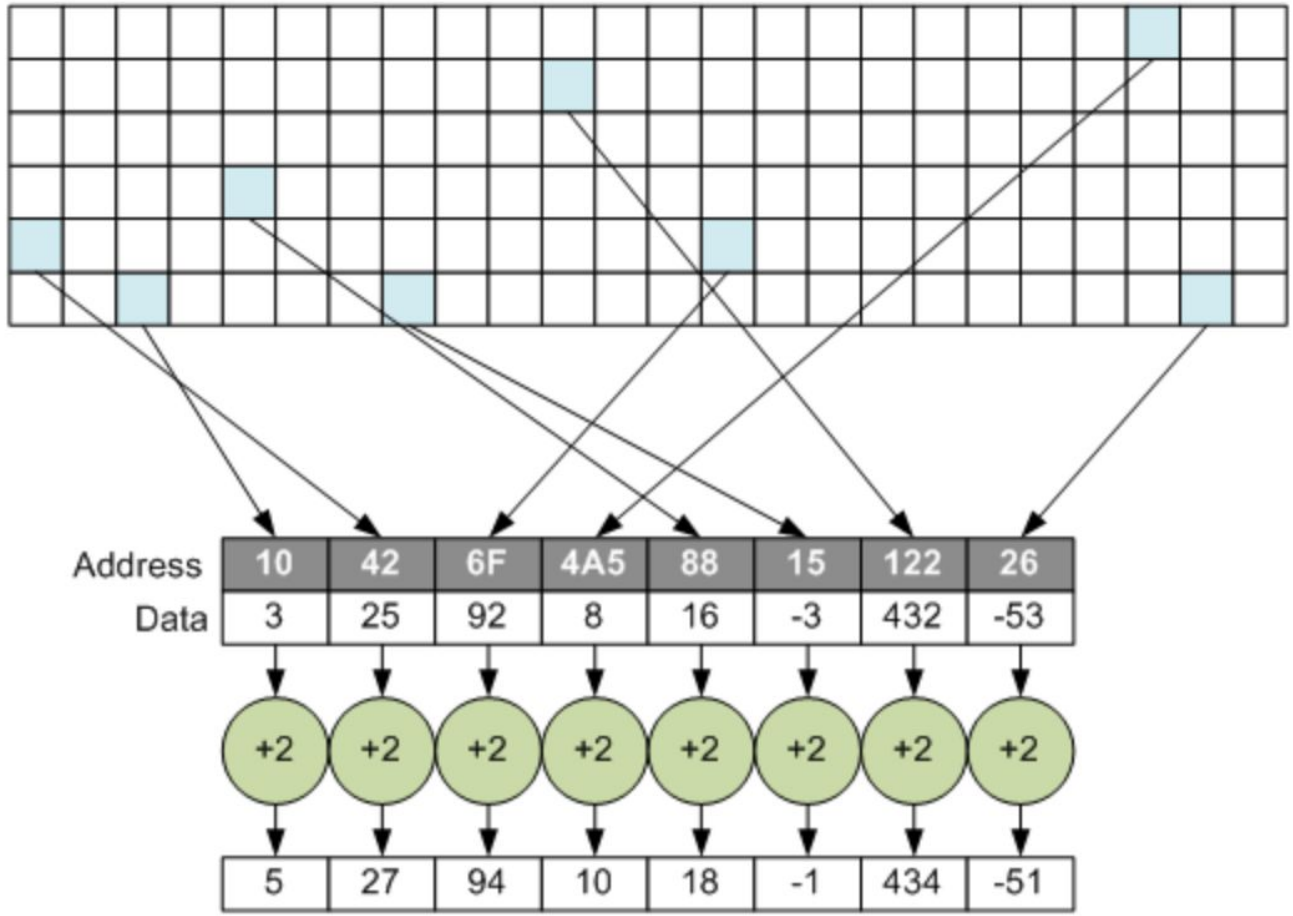
Allocation Unit[1]

- Reads two micro-ops from the instruction queue at a time
- Assigns the following resources required by the micro-ops
  - Reorder buffer entries (72)
  - Rename buffer entries (72)
  - Store data buffers (16)
  - Gather-scatter table entries (4)
  - Reservation station entries
- All these are familiar except gather-scatter table entries
  - Large vectors of data are ideally stored in contiguous memory, but this cannot be guaranteed
  - Gather-scatter entries point to blocks of memory to be used as buffers for data that is scattered in memory

Ideal[5]


DRAM
Address
Data
10
11
12
13
14
15
16
17
3
25
92
8
16
-3
432
-53
+2
+2
+2
+2
+2
+2
+2
+2
5
27
94
10
18
-1
434
-51

Scattered Data



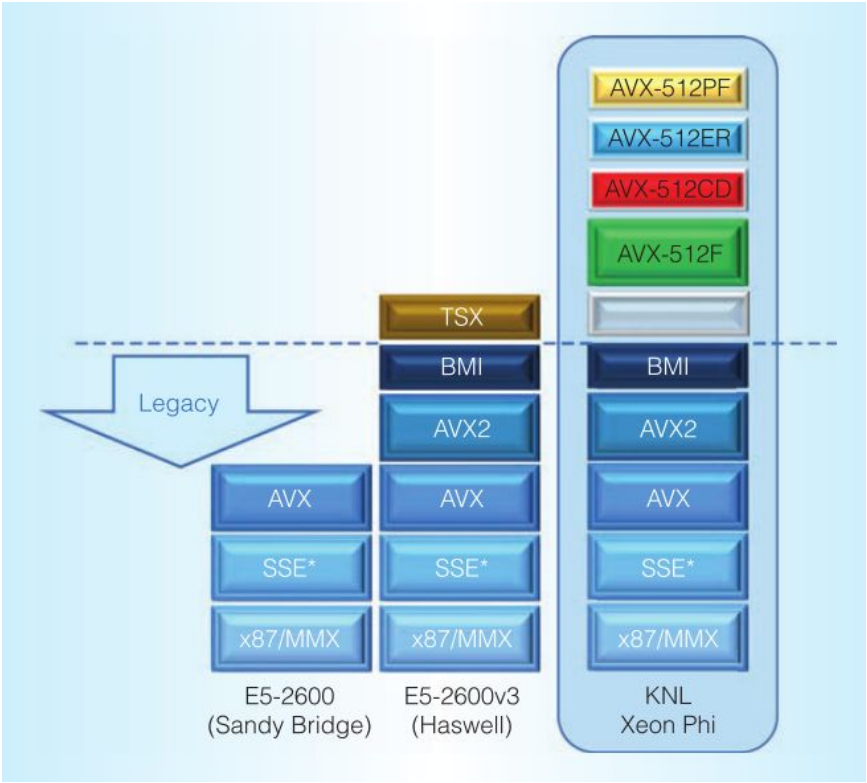
Integer Execution Unit[1]

- Each core has 2 IEUs
    - One can execute any integer micro-op, the other can only execute one-cycle latency instructions
- IEUs operate on 16 general-purpose registers
- Each IEU has a 12-entry reservation station that issues one micro-op per cycle
    - Fully out of order scheduling

Vector Processing Unit

- Each core has 2 VPU’s to handle floating-point and integer divides
- Together, they can provide a peak of 64 single-precision or 32 double-precision floating point operations per cycle
- Each VPU contains a 20-entry reservation station that, like the IEU, issues one micro-op per cycle out of order
    - One slight difference is that the reservation station entries for the VPU do not store source data

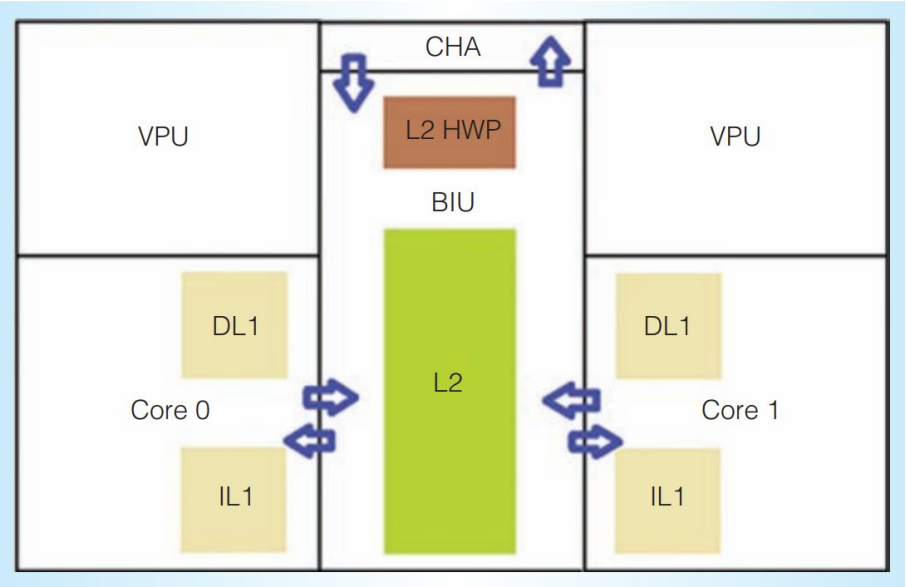
Instruction Set Architecture Improvements



- AVX-512: 512-bit SIMD instruction
  - 8x 64-bit INTs
  - 16x 32-bit INTs
  - 8x double precision FPNs
  - 16x single precision FPNs
- AVX-512F: foundation
- AVX-512CD: conflict-detection
- AVX-512ER: exponential and reciprocal
- AVX-512PF: prefetch

Memory Hierarchy

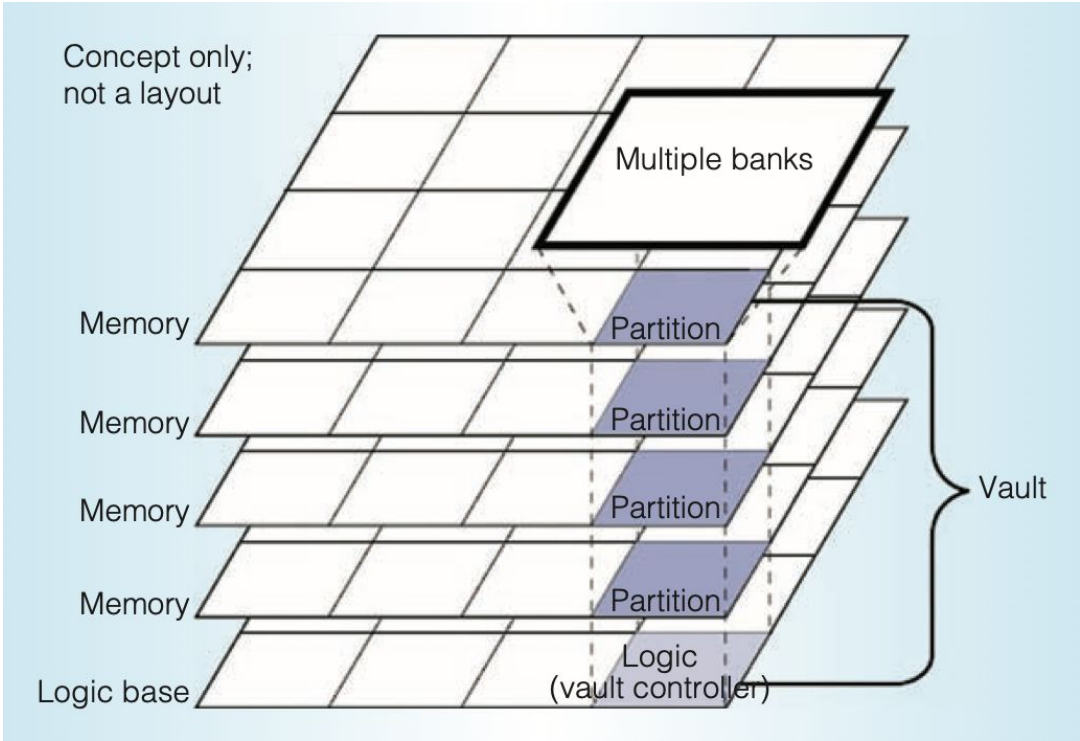
- Memory execution unit
  - issued in-order, execute and complete out of order
  - recycle buffer
  - supports unaligned memory accesses
  - contains specialized logic



Two levels of memory

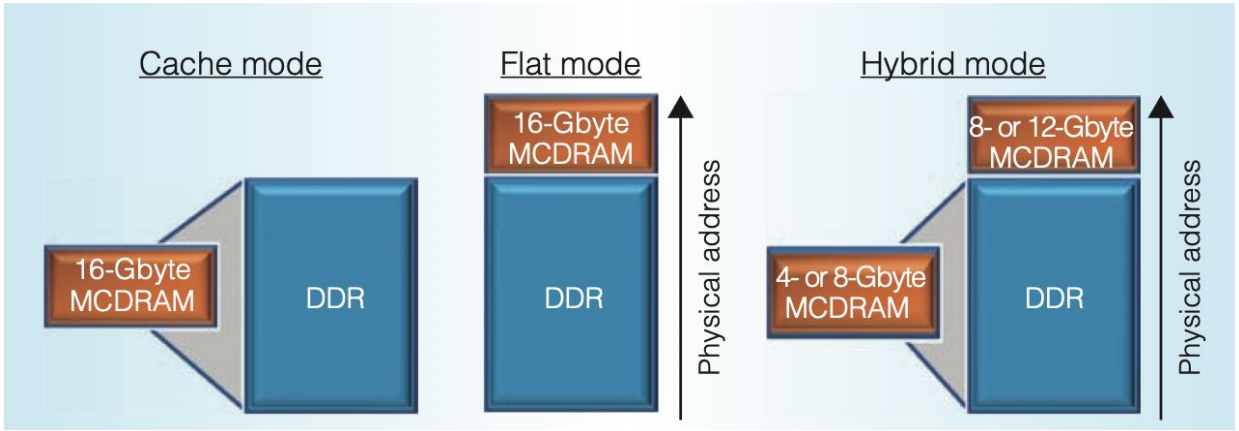
- MCDRAM: multi-channel DRAM
  - a stacked DRAM architecture
  - provides many more channels

- DDR: Double Data Rate memory



Three memory modes

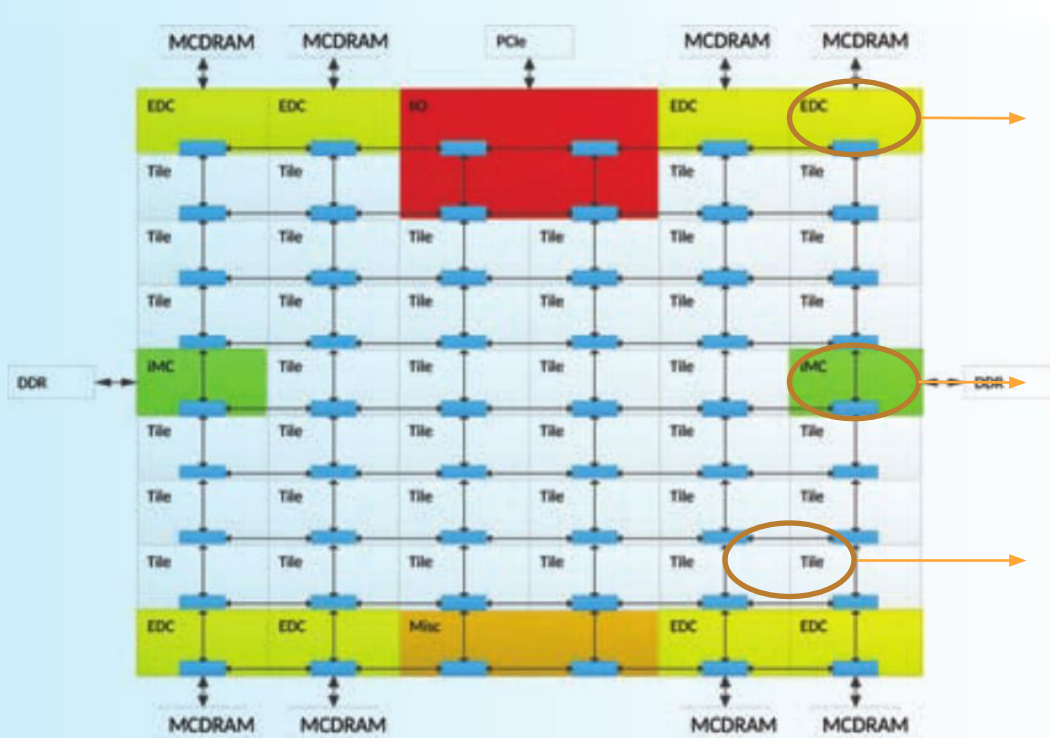
- Cache mode
  - MCDRAM is configured as a memory-side cache for the entire DDR memory
  - Modified, exclusive, garbage, invalid protocol
- Flat mode
  - regular memory mapped in the same system address space
- Hybrid mode
  - split either half or a quarter of the MCDRAM as cache; the rest is used as flat memory



Threading

- Supports up to four hardware contexts or threads per core
- Resources are divided using 3 techniques
  a. Dynamic Partitioning
     - ROB
     - Rename Buffer
     - Reservation Stations
     - Store Data Buffers
     - Instruction Queue
     - Gather-scatter table entries
  b. Shared Resources
     - TLB
     - Branch Predictor
  c. Replication
     - Rename Tables

Interconnection Network

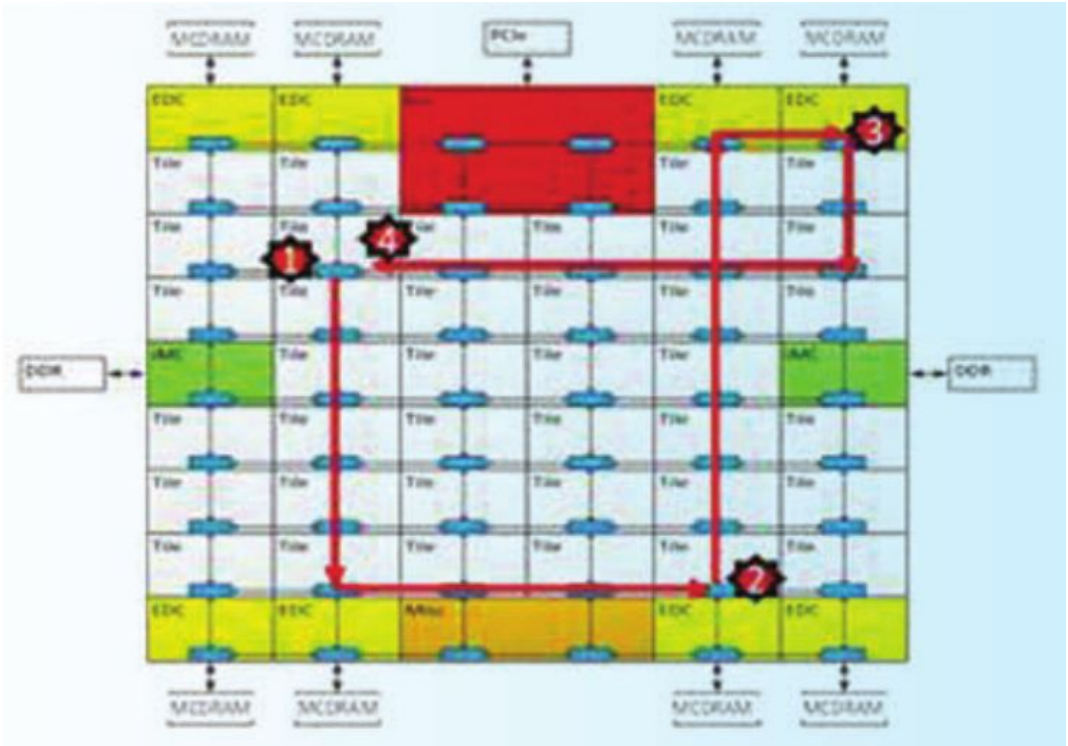


Mesh Architecture

Cache Consistency Model[3]

- MESIF Protocol:
  - Modified, Exclusive, Shared, Invalid, Forward
  - No Owned State (dirty cache block)
- Forward State (specialized S state):
  - Clean cache block
  - Directly forward shared data instead of going to memory
  - Directly forward shared data instead of requesting all the shared caches
  - *Can co-exist with owned state

Mesh Routing (YX Routing Rule)[1]



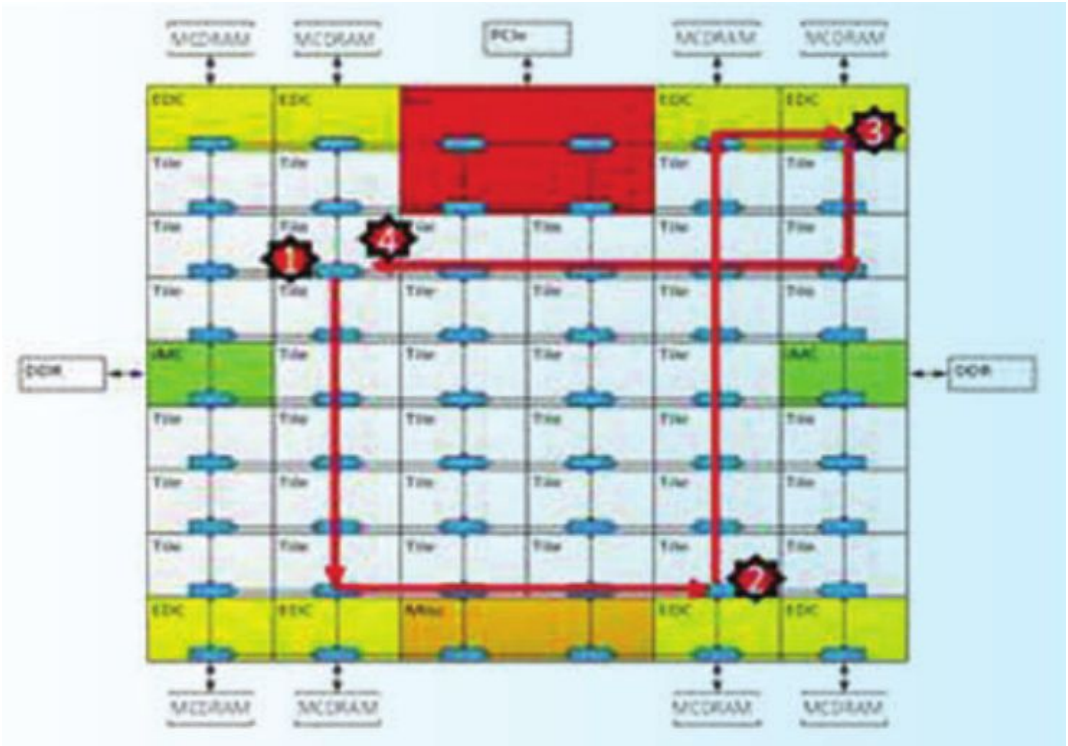
Benefits:

1. Reduce deadlock cases
2. Simplify the protocol

Mesh Cluster Modes

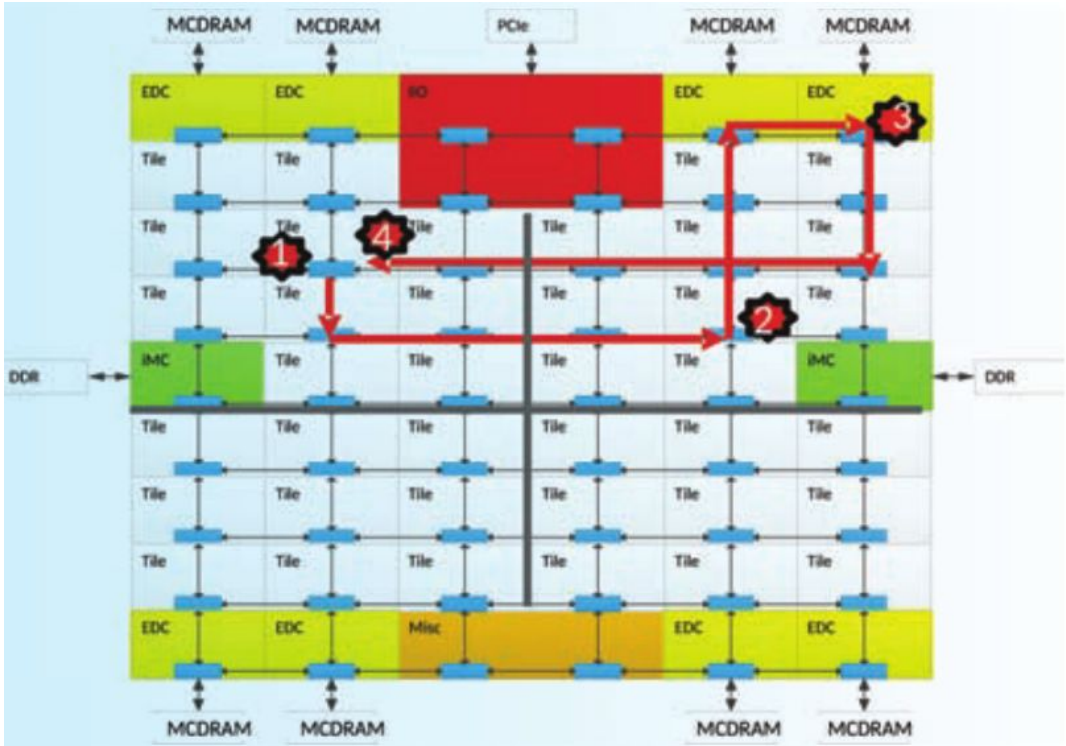
- Different levels of MEM address affinity to improve performance
- Affinity: tiles, directories and memories
- Local cluster access -> lower latency & higher throughput
- *Selectable from BIOS at boot time

All-to-all Mode



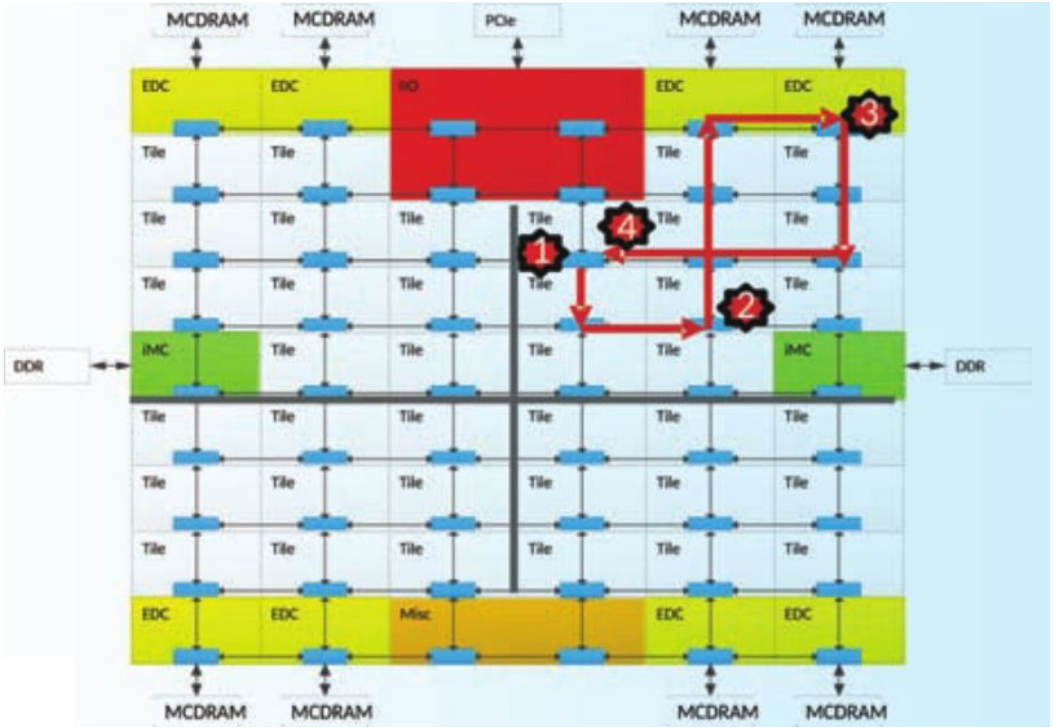
- No affinity
- Most general mode
- Lowest performance

Quadrant Mode



- Affinity between directory and memory
- TD accesses MEM in the same quadrant
- Default mode
- Required: symmetric MEM

Sub-NUMA Clustering (SNC)



- NUMA: non-uniform memory access
- Expose chip as 2/4 NUMA domains
- Affinity among tile, directory and memory
- Best performance if APP is NUMA-aware

References Slide

[1] Sodani, Avinash, et al. "Knights landing: Second-generation intel xeon phi product." *Ieee micro* 36.2 (2016): 34-46.
[2] Michaud, Pierre, André Seznec, and Richard Uhlig. "Trading conflict and capacity aliasing in conditional branch predictors." *ACM SIGARCH Computer Architecture News*. Vol. 25. No. 2. ACM, 1997.
[3] Hum, Herbert H. J., and James R. Goodman. Forward State for Use in Cache Coherency in a Multiprocessor System. 18 Mar. 2003.
[4] Rahman, Rezaur. “Intel® Xeon Phi™ Core Micro-Architecture.” *Intel*, Intel, 15 Oct. 2019, software.intel.com/en-us/articles/intel-xeon-phi-core-micro-architecture.
[5] Moyer, Bryon. “How Does Scatter/Gather Work?” *EEJournal*, 9 Feb. 2017, www.eejournal.com/article/20170209-scatter-gather/.