

Introduction to On-chip Network and Cache Coherency

By Kishore Mishra, President and CTO

The recent trend in computer architecture is to use multiple CPU cores inside a chip to scale up performance and reduce latency. A program or task can be broken down to independent steps that can be executed in parallel by multiple cores. When the cores need to share data or control information, they communicate with one another through an on-chip network. With large number of cores, an efficient on-chip network is used as communication medium. There are different ways a network can be formed, such as ring or mesh (matrix) network.

One of the challenges in multi-core design is how to efficiently share data among processors. One way is to use the main memory (DRAM) as the deposit house for all cores to share data. Since main memory is off chip, accessing it not only takes an order of magnitude longer compared to staying within the chip, it also consumes a lot of power. One solution currently used is to have a local L2 cache in each CPU core that is accessible by all other on-chip processor cores as well. The idea is to share the data within the chip and postpone trip to memory as long as possible. This not only speeds up the inter-processor communication, but saves power as well. However, there is a big challenge maintaining coherency among the cores. Coherency, in simple term, means any processor should get the latest data, not old or stale data. Next, we discuss how this problem is currently addressed in the industry.

To maintain coherency and share L2 cache there are two types of architectures used today – snoop-based architecture and directory-based architecture. In snoop-based architecture, message is broadcast to all CPU cores, and every core takes necessary steps to keep up-to-date. Snoop-based architecture is simple to implement, but is not efficient in terms of performance. For smaller number of core counts (up to 6-8), snoop-based architecture could be a good choice. For larger number of core counts (typically more than 8), directory-based architecture is used to scale up performance.

At the heart of directory-based cache-coherent architecture is the concept of home node where a directory controller manages the communication among processors. All cores send the requests to their assigned home nodes, and the directory controller in the home node manages the show. It serializes requests from all cores and executes them in serial order, and this is how coherency is maintained. Before we go into more details, let us discuss how the memory addressing is done. The memory space can be divided into a coherent and non-coherent space.

The processors or other devices that share data in the coherent address space, communicate to the directory instead of talking to one another directly. The coherent memory space is further divided into units of memory called cacheline (typically 32, 64, or 128 bytes of data).

The coherent memory address space is divided into multiple chunks among the home nodes (there could be one or more home nodes in an on-chip network). Each home node (refer Fig 1) is in charge of part of the non-overlapping coherent memory space. When a CPU core wants to access a cacheline in coherent space, it sends the request to the network, and the on-chip network forwards the transaction to the home node that owns the address space belonging to the cacheline.

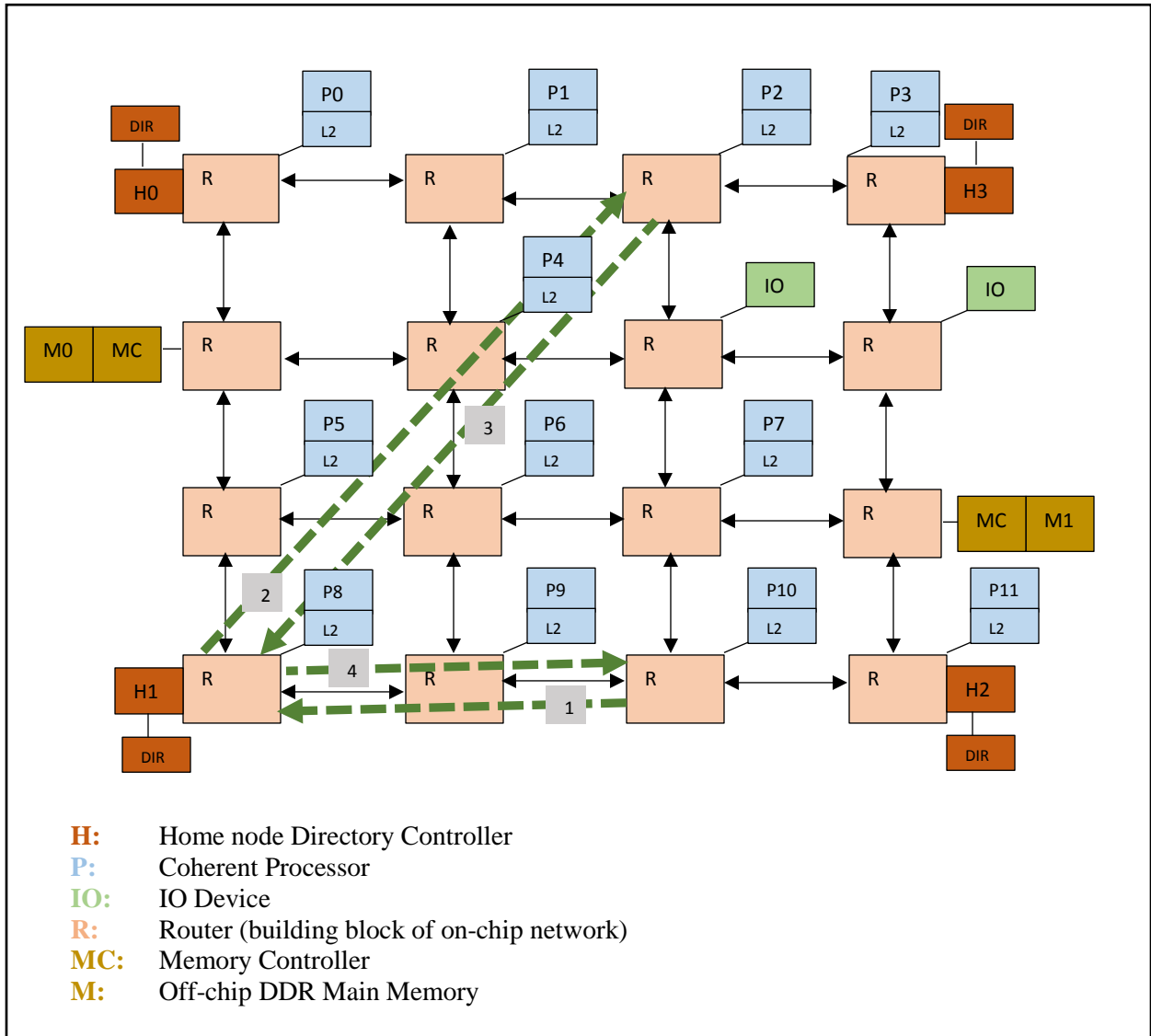


Fig1 4x4 On-chip Mesh Network

The directory controller in each home node maintains details related to each cacheline, such as which cores have a copy of the cacheline, the current cache state (valid, unique, shared, clean, dirty) etc. Let's have a walk thru where one of the processor cores (P2) has the data in its L2 cache, and another processor core (P10) wants to get a copy of the data to its L2 cache.

- Step1:** P10 sends the request to home node H1
- Step2:** H1 sends the request to P2 which has the latest copy of the data
- Step3.** P2 sends a copy of data to home node H1
- Step4.** H1 sends the data to the requesting core P10.
- Step5.** Requesting core P10 sends an acknowledgement message to home node H1 indicating that it received the data.

At this point the original request to that cacheline from core P10 is complete, and home node H1 can start executing another request to the same cacheline. The advantage of directory-based architecture comes from the ability to send directed messages to cores instead of broadcasting to all the cores on the network. This reduces traffic on the network significantly which allows larger number of cores to be on the network.

The basic building block of an on-chip network is a router that connects to one node (CPU, IO, or memory) and 4 other neighboring routers in North, South, East, and West directions. The routers can be connected to form a ring topology or two dimensional MxN mesh topology. There are other three dimensional topologies such as Torus and hypercube that can be formed by using the basic router. However, three dimensional topologies are not suited for on-chip network. In a ring topology, it is easier to physically connect and place the routers inside a chip as each router needs connection only in two directions (left/right or top/bottom). However, it is not suitable for large number of nodes as the worst-case latency grows linearly with number of nodes. Two dimensional mesh topology (nodes arranged in an x, y grid) provides a better alternative for larger number of nodes on the network. For example, in a 6x6 interconnect mesh network (refer Fig 2), the worst case latency between any two nodes is (10 hops), whereas it would need 18 hops when 36 nodes are connected in a ring topology.

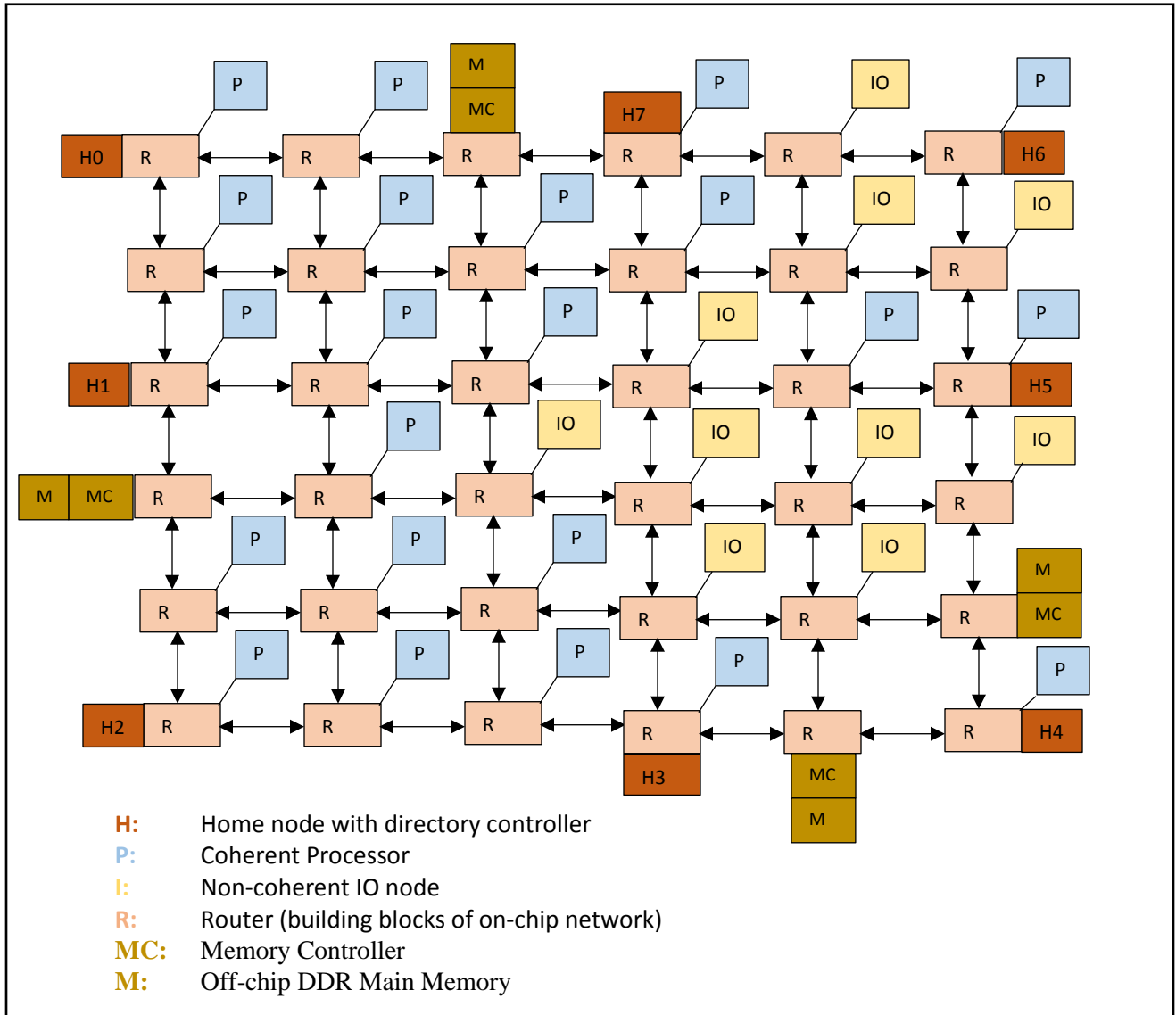


Fig2 Example 6x6 On-chip Mesh Network with 36 nodes