Multi-Core Computer Architecture: Storage and Interconnects Week 6

Aronya Baksy

October 2021

1 Introduction to Tiled Chip Multi-Core Processors

- Traditional multicore processors have a bus-based mechanism wherein all the cores communicate with each other and with the main memory via buses
- The drawback here is that the bus availability becomes a performance bottleneck, especially when all cores are busy (not scalable, not power efficient, not performant beyond a limit)
- The paradigm shift is away from bus-based architecture, to a tiled-chip interconnect-network based architecture

1.1 Network on Chip (NoC)

- Each processing unit is called a tile. Each tile has a 2D index (X, Y)
- Tiles are interconnected by a network of routers. Tiles communicate with each other by sending packets
- Packets are divided into flow control units called **flits**. L1 and L2 cache misses create packet traffic on network
- Each tile has a private L1 cache. The shared L2 cache is split among all the tiles
- Each tile is connected to its north, south, east and west neighbours

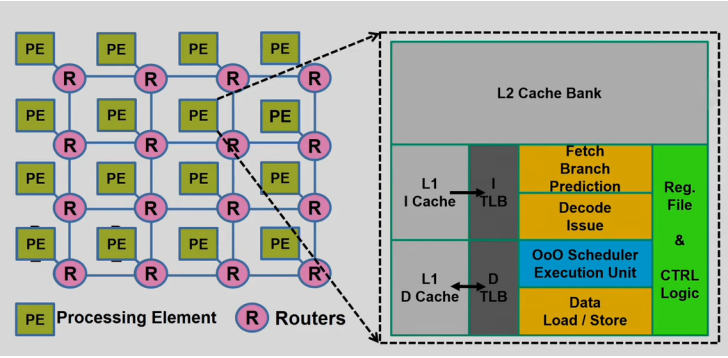

Figure 1: Tiled Chip Multi-core Architecture

1.2 Cache Address Mapping in TCMP

- The address is split into tag, processor, index, block offset
- Once a cache miss is encountered in one core, it requests the data in packets from the core where the address maps to
- Packet is a basic unit of transfer for the network. Flits are sub-units of packets, which are the basic unit of transfer between 2 routers
- All packets consist of a head flit, the body flits and a tail flit.

2 Building Blocks of TCMP

2.1 Topology

- Physical layout and connection patterns between nodes and channels in the network
- e.g.: ring, star, spidergon, 2D mesh, 2D torus

2.1.1 Mesh Topology

- Each node connected to N, E, S, W neighbours (see 1)
- Easy to layout on chip (regular and equal length links)
- Path diversity: many paths between 2 nodes
- Not symmetric. Sensitive to placement of task on an edge node vs a middle node

2.1.2 Torus

- Torus solves the asymmetry of the mesh (here every node, even the edge ones, have 4 neighbours)
- Harder to layout on chip and unequal link lengths

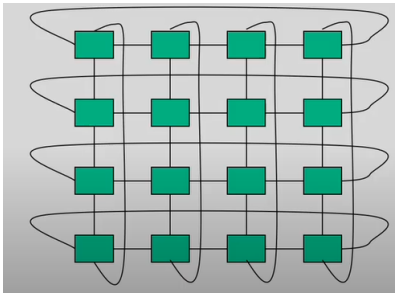

Figure 2: Torus Topology

2.1.3 Hierarchical Rings

- More scalable and lower latency, but more complex to set up
- On top: with one bridge router per smaller ring, two bridge routers per smaller ring, and 4
- Bottom: Reusing the 2 bridge router setup to extend to 64 nodes.

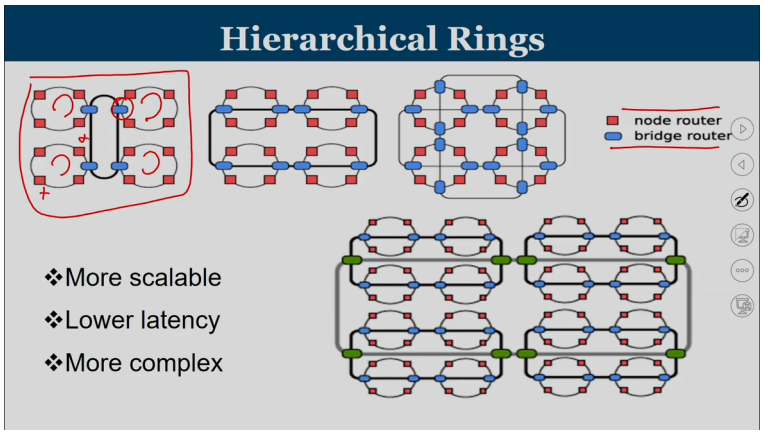

Figure 3: Hierarchical Rings of different types

2.1.4 Tree

- Planar hierarchical topology (H tree)
- Good for local traffic, easy to lay out on chip
- Root becomes a bottleneck (fat trees, CM5 solve this issue)

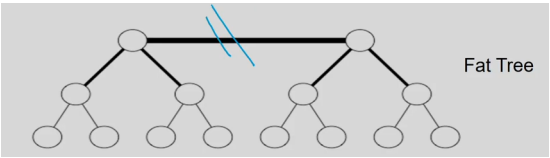

Figure 4: Fat Tree

2.2 Routing Algorithm

- Find optimal path between source and destination nodes
- Classification of routing algorithms:
  - **Deterministic**: Always chooses same path between a given source and dest. pair
  - **Oblivious**: Chooses different paths, without considering the network state
  - **Adaptive**: Choose different paths based on the network state (congestion status)
  - Minimal vs non-minimal: A **profitable** route is one that leads a packet closer to its destination. A minimal routing algorithm chooses a profitable route. Non-minimal algo may not do this
  - Source routing vs Node routing:
    - In source routing, the route is known from source and embedded in packet header (reduces packet processing delay, but routing table stored in each node).
    - In Node routing, each node stores the next turn to be taken for the particular destination. At each router the next turn is checked. Can incorporate dynamic conditions
  - Deadlocks are caused by circular dependency on resources. Each packet waits for the downstream buffer to be empty in a loop
  - Deadlock halts all forward progress in routing
  - Deadlock handling: avoid cycles in routing, add more buffers, detect using pre-emption
  - Deadlock avoidance: avoid certain specific turns (e.g. west first and north last, negative first)

2.2.1 Deterministic Routing

- **Dimension order routing**: traverse X first then Y. Only 1 90 degree turn.
- Free from deadlock, but high contention and does not exploit path diversity

2.2.2 Adaptive Odd-Even Turn Routing

- BAsic ideas: prohibit certain turns at certain junctions
- For nodes in even columns, east->north and east->south turns are not allowed
- For nodes in odd columns, north->west and south->west turns are not allowed
- Note that all columns are numbered from 0 and 0 is an even column

2.2.3 Minimally Adaptive Routing

- Router uses downstream network state (e.g. buffer occupancy) to pick the productive output port
- This algo is always aware of local congestion
- Restricts load balancing and link utilization

2.2.4 Non-Minimal Fully Adaptive Routing

- May misroute packets to non-productive outputs based on the downstream network congestion
- Can achieve better load balancing and link utilization

2.2.5 Contention Look Ahead Routing

- Foresee contention between neighbours using control signal wires
- These control wires (channels) are used to obtain traffic condition of neighbours

2.3 Flow Control

2.3.1 Handling Contention

- Two packets trying to use one link at the same time: buffer one (common), drop one, misroute one

2.3.2 Store-and-Forward

- The incoming packet is copied entirely into the router before forwarding it to the next node
- Here the packet is the basic unit of flow control
- Leads to high per-packet latency, buffer space for entire packet in node

2.3.3 Cut-Through Flow Control

- Start forwarding as soon as the head flit is received and the connection resources (buffer, channel) are allocated
- Reduces latency
- If output port for head gets blocked, then tail continues when head is blocked, thus absorbing the full message into one switch (hence buffer space needs are more)
- With high contention, it degenerates to store-and-forward

2.3.4 Wormhole Flow Control

- Packet broken into flits. Body follows head, tail follows body and this happens in a pipelined manner. Only the head flit contains the routing info
- If head is blocked then rest of the packet is blocked too
- Lower latency and efficient buffer utilization but occupy rsources across multiple routers

2.3.5 Virtual Channel Flow Control

- Multiplex virtual channels over a single physical channel. Divide input buffer into multiple buffers sharing a single physical channel
- The virtual channel is allocated once to the head flit. Remaining flits of packet use the same VC.
- Avoid deadlocks using VC

2.3.6 Functions of Router

- Buffering of flits
- Route computation
- Virtual channel allocation: reserving buffer space in the downstream router
- Switch allocation (between multiple flits to same o/p port) and traversal
- Link traversal (not in router)

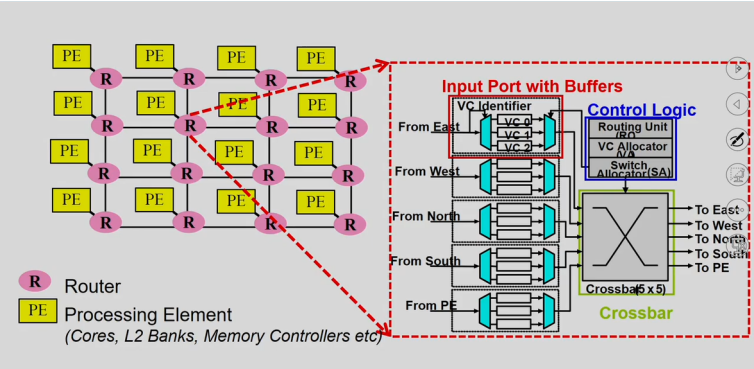

Figure 5: Input-Buffered NoC Router

2.3.7 5 stage Router Pipeline

- BW: Buffer Write
- RC: Route Computation
- VA: Virtual channel Allocation
- SA: Switch Allocation
- ST: Switch Traversal
- RC and VA is only done for head flit of each packet
- Pre-computing route for next buffer allows flits to compete for VCs immediately after buffer write. (RC decodes route header)
- Speculative routing: assume that VA happens successfully, and hence perform VA and SA in parallel.
- If VA fails, then repeat both steps again

2.3.8 Selection Strategy

- Adaptive routing function returns set of possible output channels (SPOC)
- Output selection function takes SPOC and neighbour feedback to give selected output channel (SOC)
- Input channel selection: from multiple flits at the input all wanting to go to the same output port of router, which one to pick?
- Output channel selection: One flit can be routed to multiple output ports. Which output to pick?
- Good selection strategy reduces avg packet latenciy