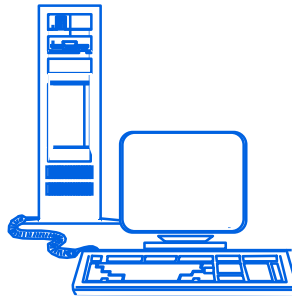


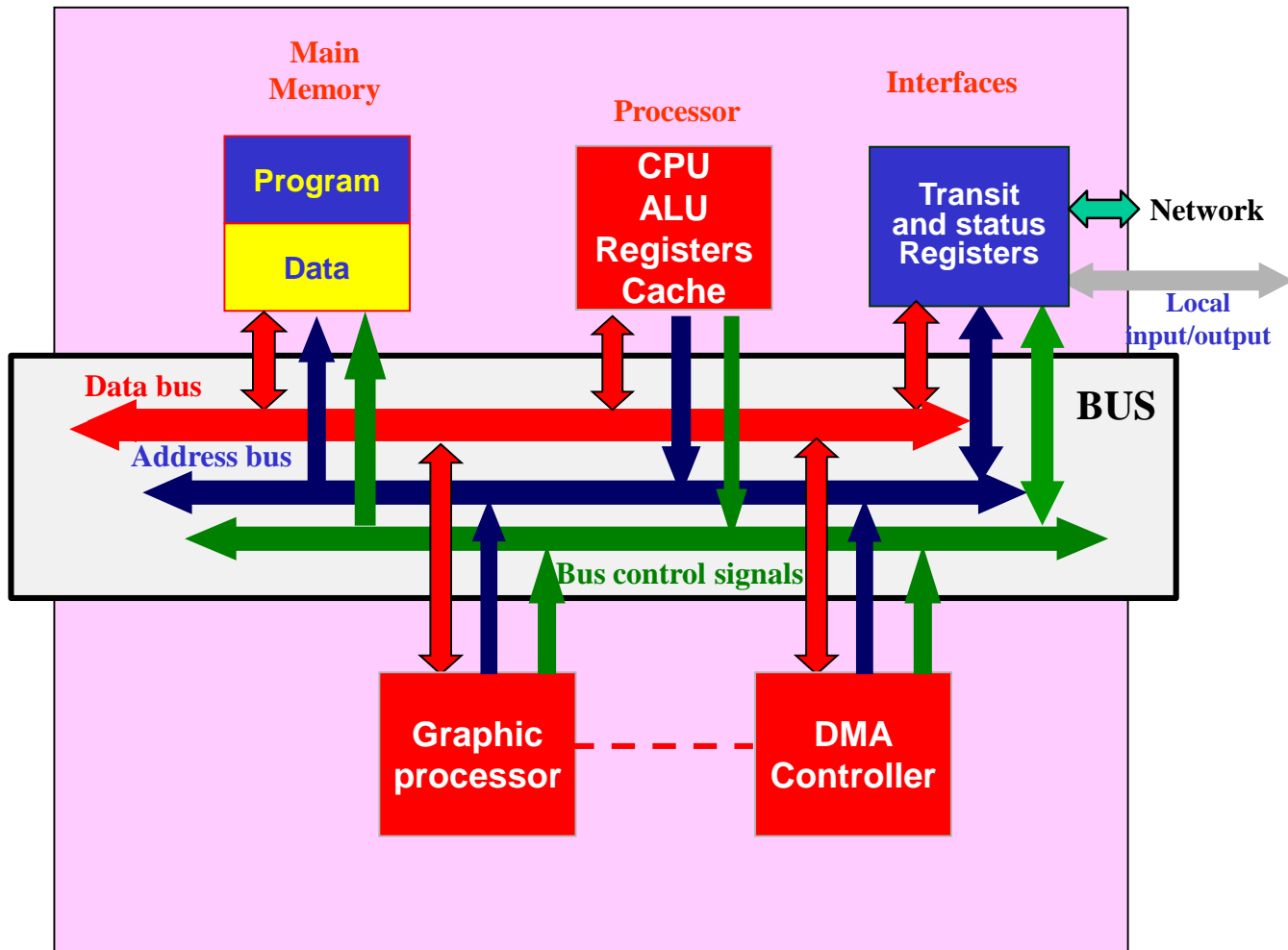
BUS

Computer Architectures M



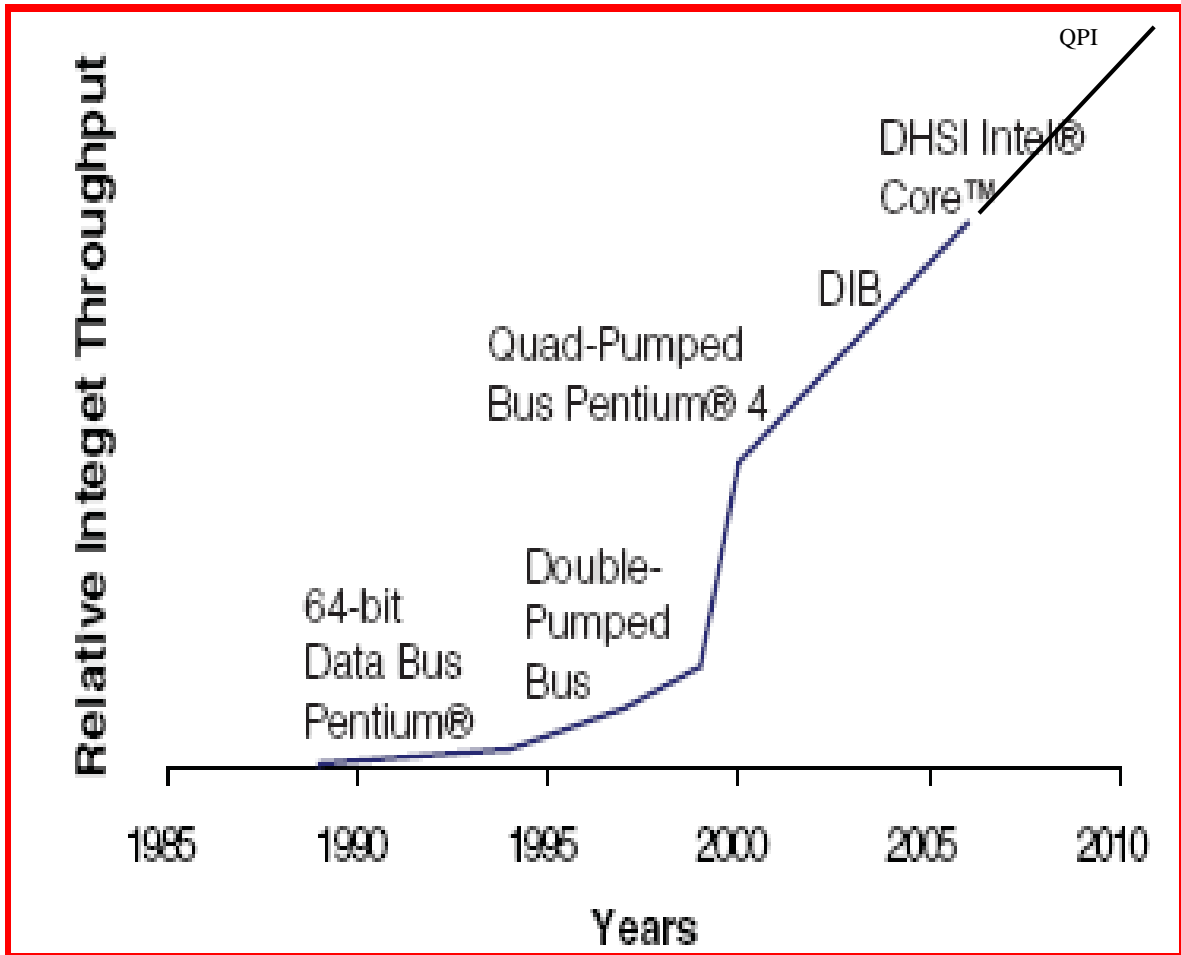
*Some drawings are from the Intel book
«Weaving_High_Performance_Multiprocessor_Fabric»*

Traditional bus



- Active agents (processor, graphic controller, DMA etc.) issue first the address (and the data on the data bus in case of write) and then pulse a line (read or write on the bus control signals) to read (or to store) the data on the data bus
- The data destination (or the source) can be either the memory OR the input/output
- Parallel bus. 64/128 address lines (how many GB?), 64/126 data lines, 30 control lines (Rd/Wr, Memory/IO, interrupts...)
- All data transfers require the bus: **BOTTLENECK**

Bus evolution



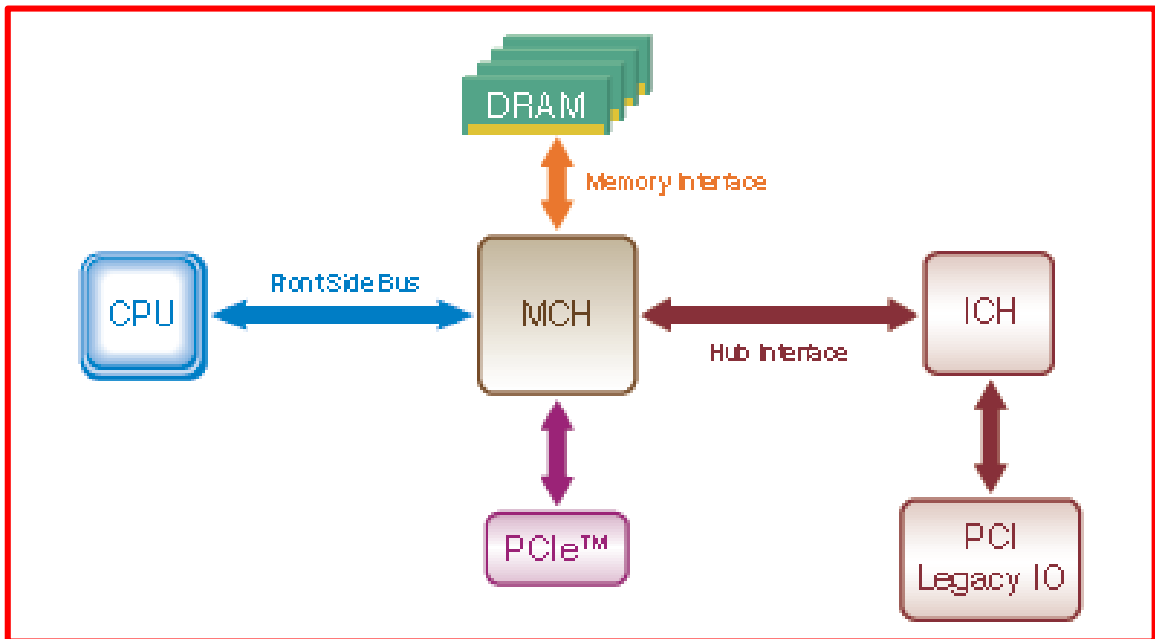
DIB: Dual Independent Buses

DHSI: Dedicated High Speed Interconnects

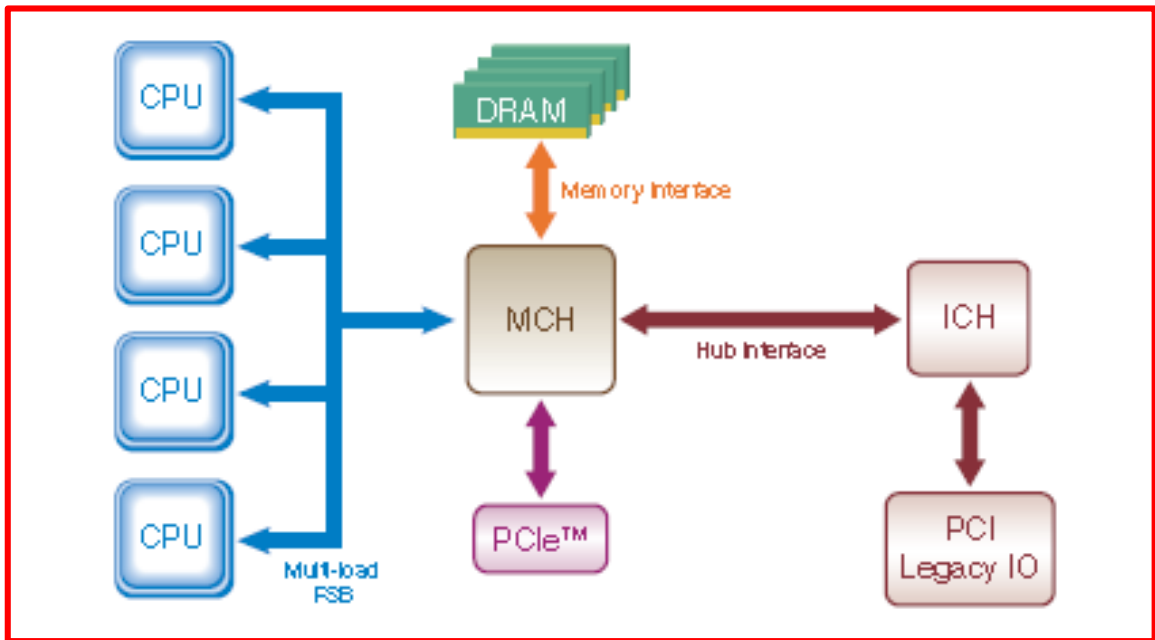
Quick Path (after 2010): bus serial evolution

- *Point to point* **packetized bus**
- **Greater bandwidth**
- **Snoop/coherency protocol**
- **Communication paths dynamically reconfigurable**

FSB until 2004



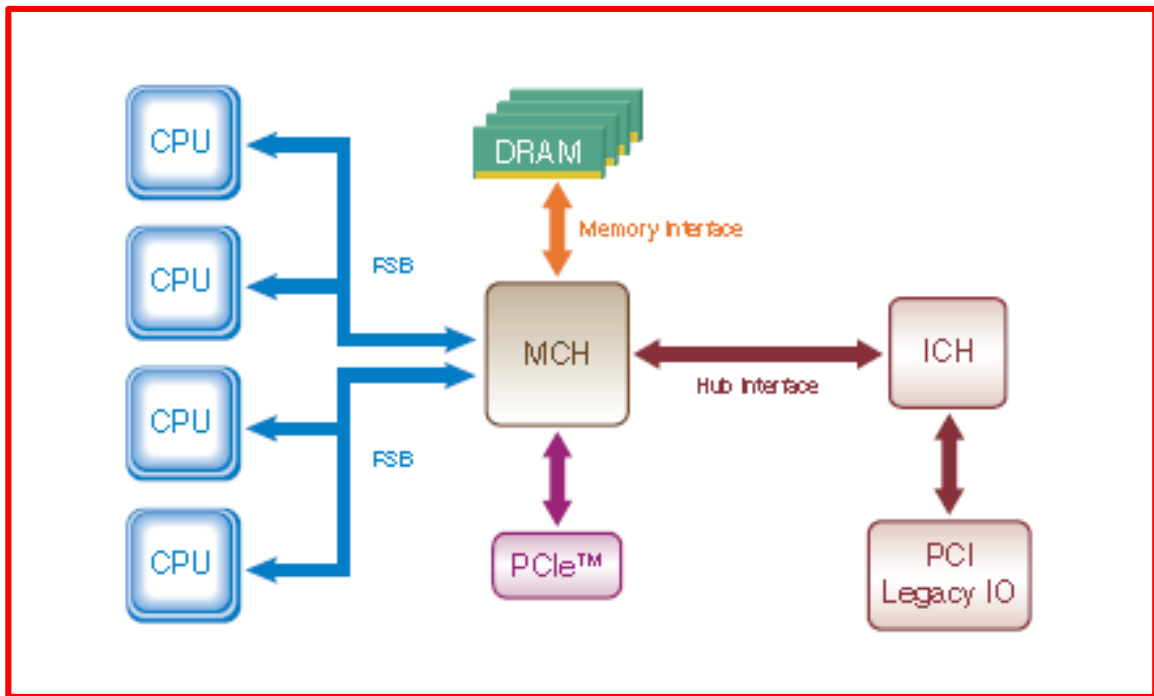
Monoprocessor



Multiprocessor

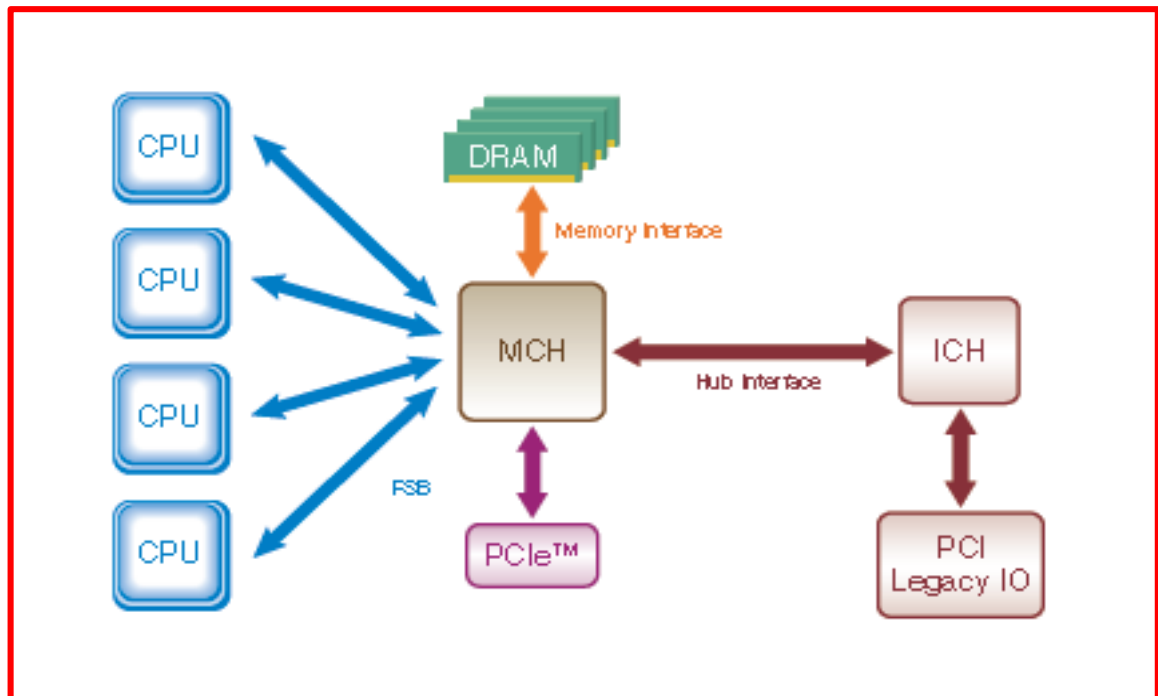
MCH Memory Central Hub
ICH I/O Central Hub

DIB (2005-2007)



Snoop traffic *must however* entail both busses

DHSI (2007-2008)



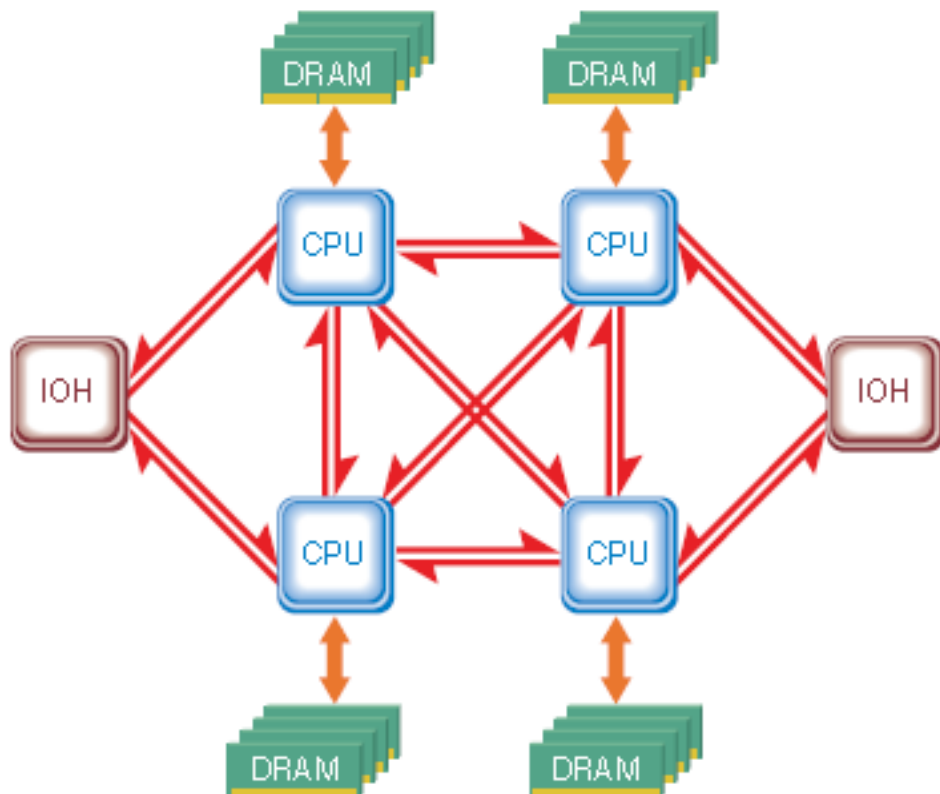
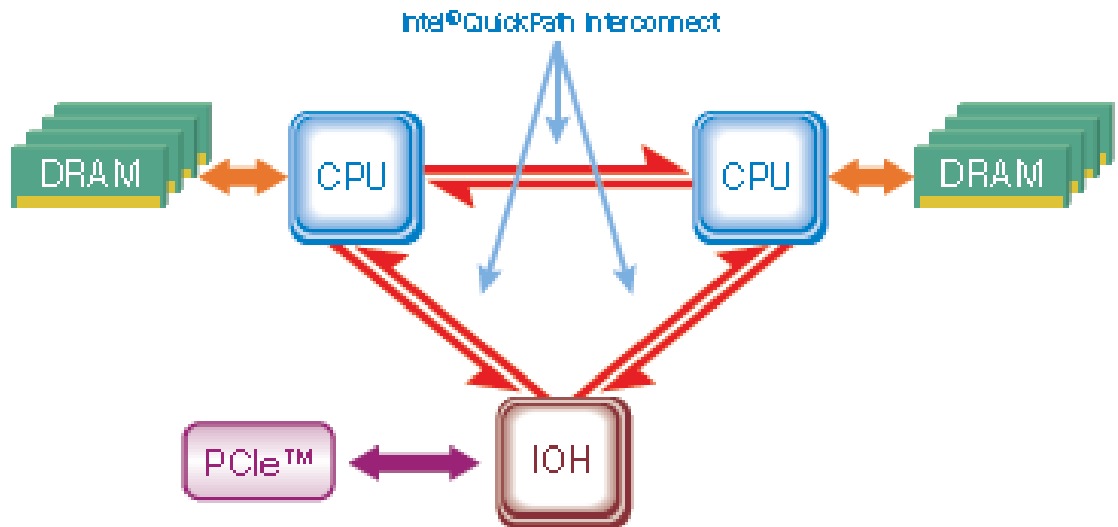
- Still snoop problems
- Centralized control
- Similar to the twisted pair Ethernet

New requirements

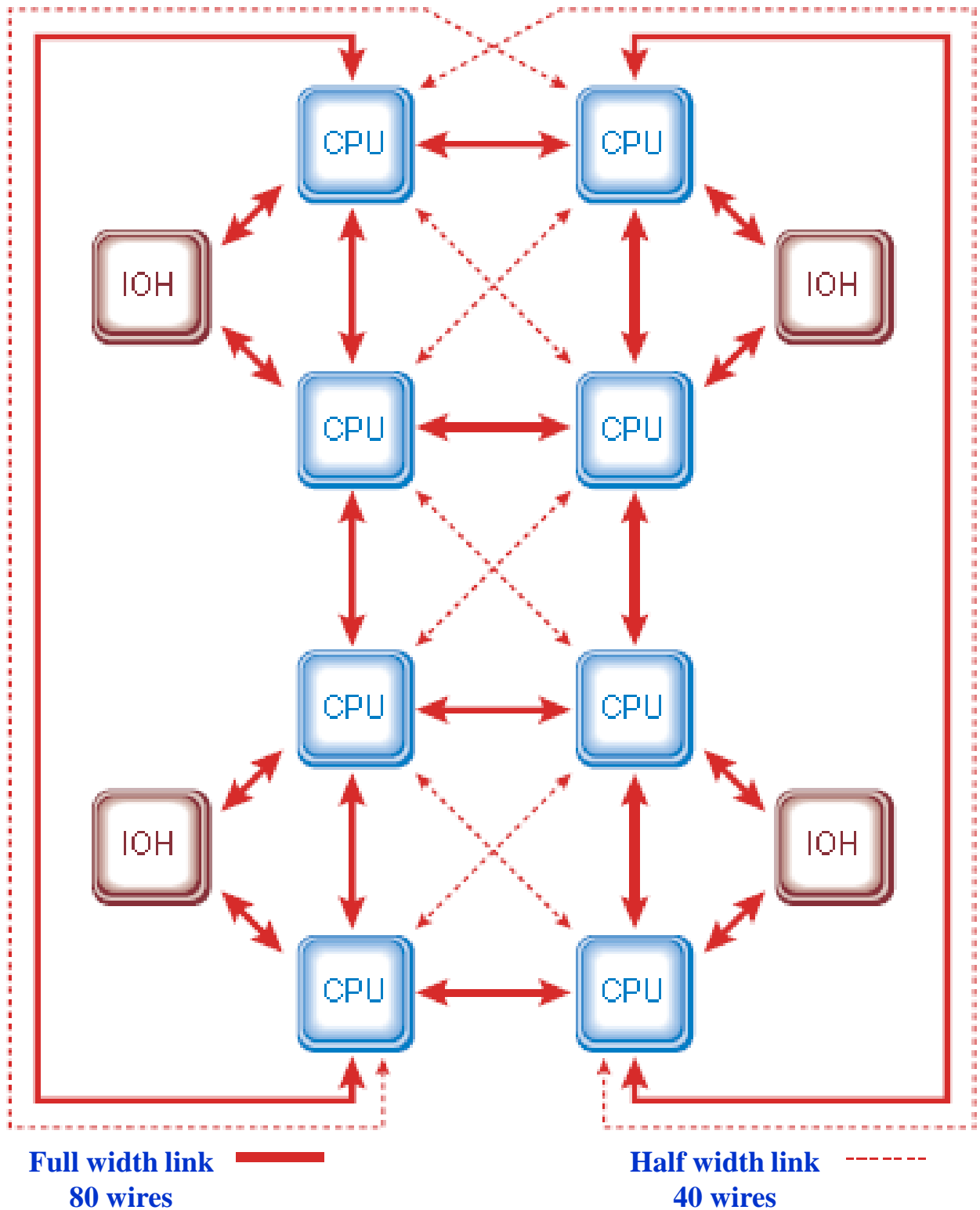
- *Bigger overall bandwidth*
- Total hw/sw transparency after initialization
- Reconfigurability and scalability
- Low costs
- Reliability
- Provision for future needs
- *Lower number of connections (wires) between blocks*

Quick Path (84 wires/connection maximum)

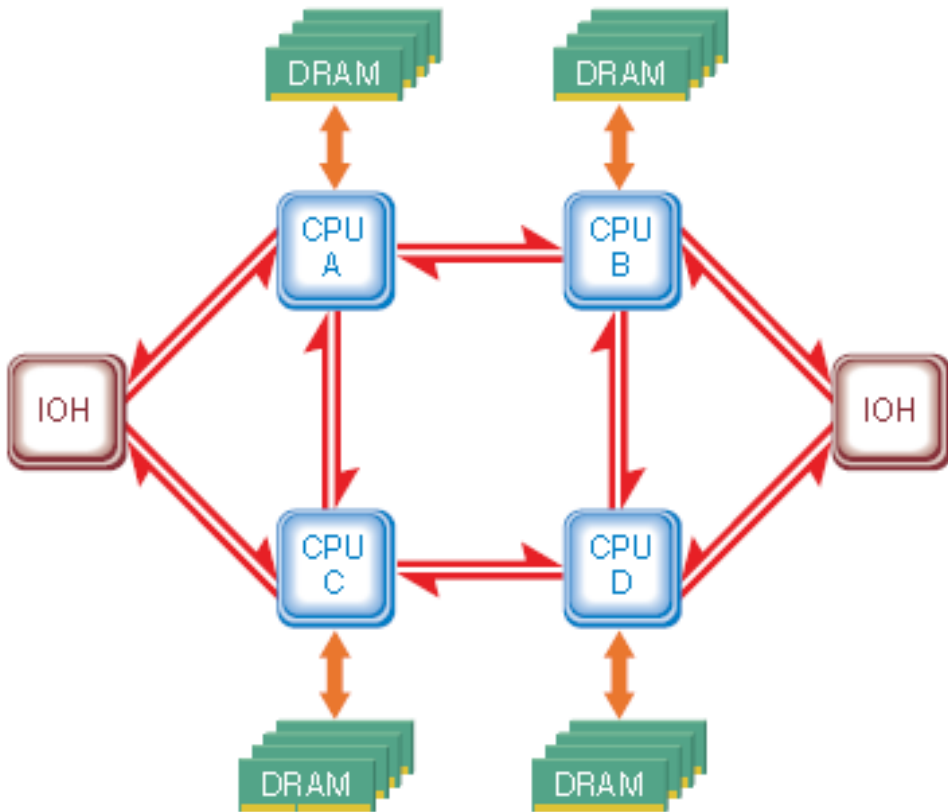
Quick Path fully connected



Quick Path

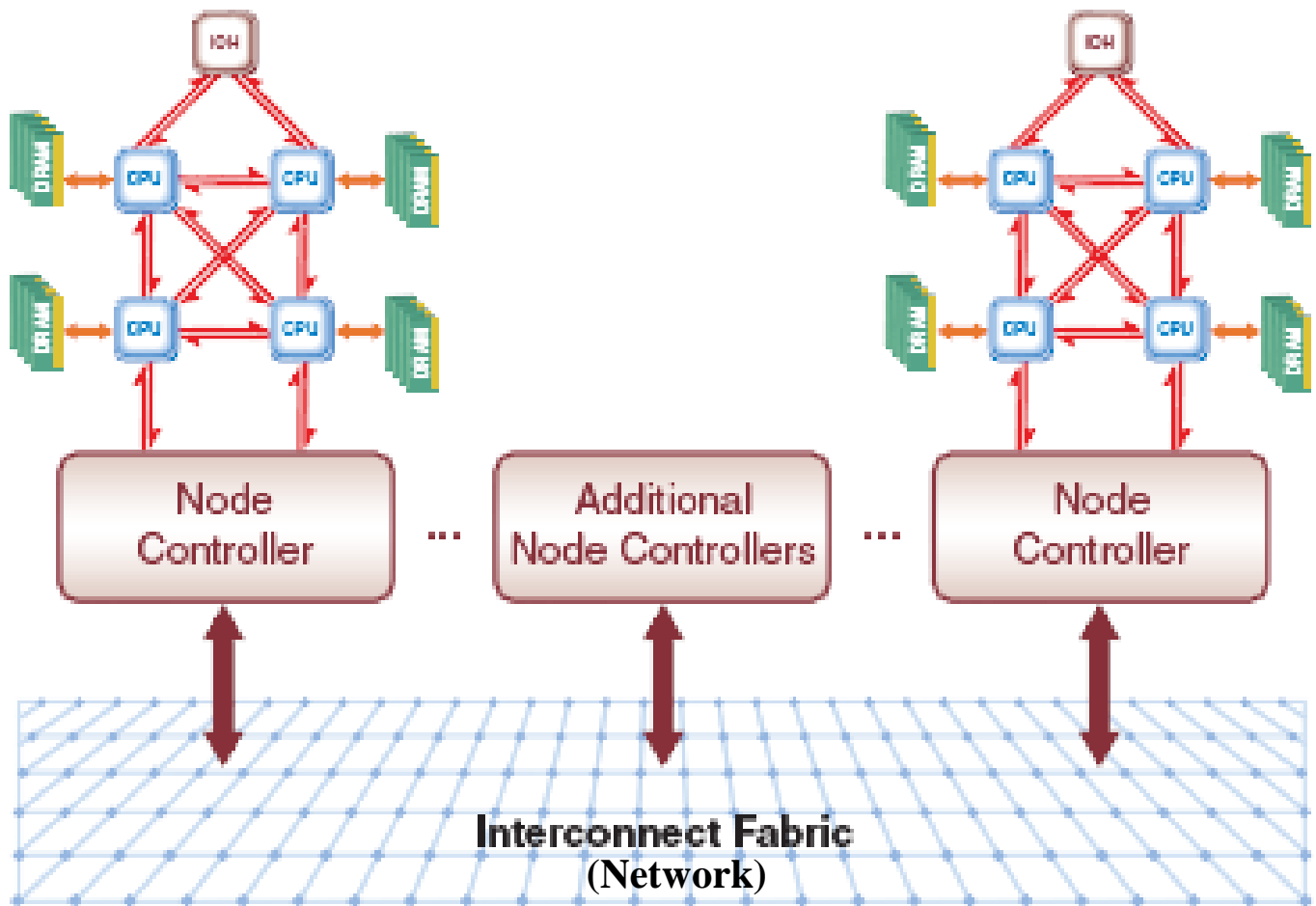


Quick Path partially connected



**In this architecture the connection between A and D –
for instance – requires the use of AC and CD
connections**

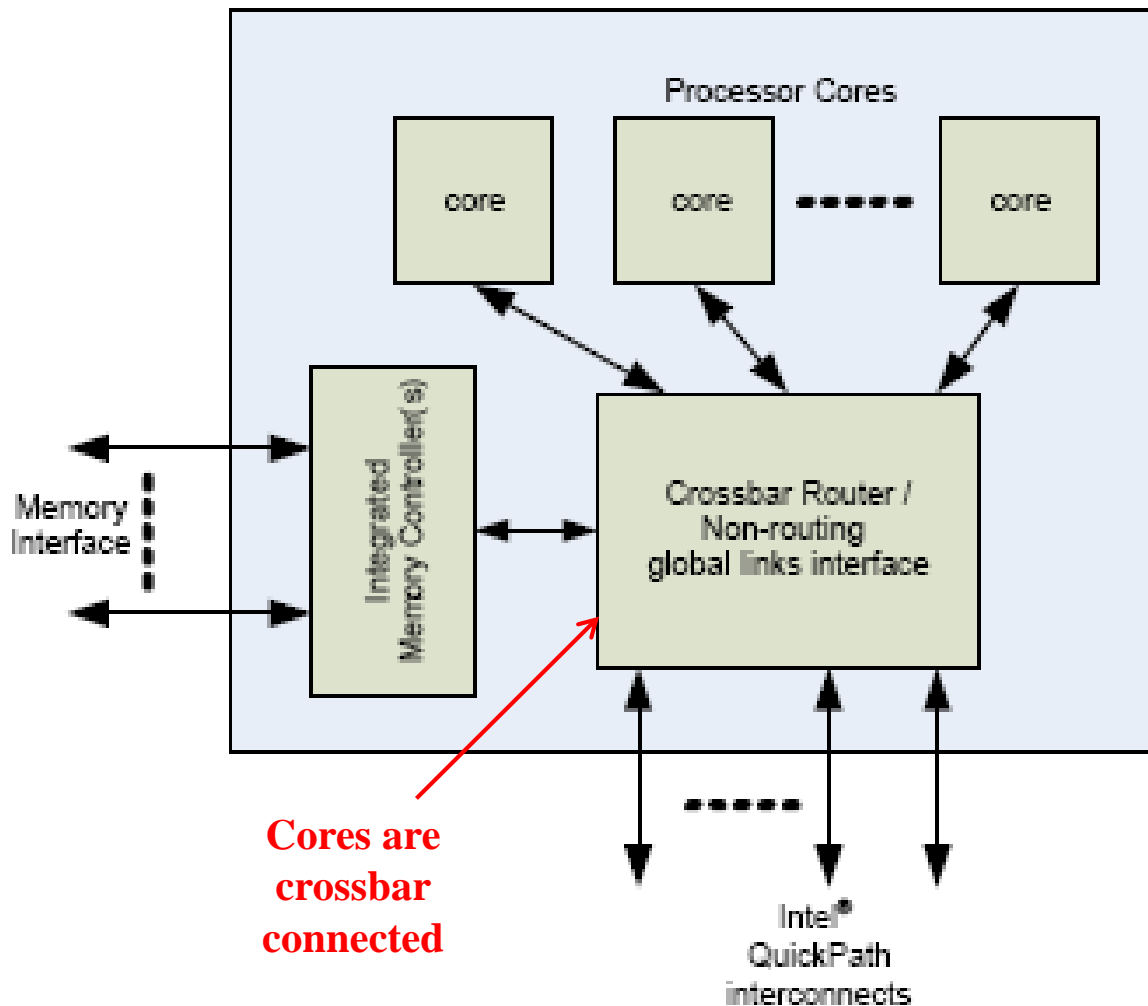
Hyerarchically connected Quick Path



Terminology

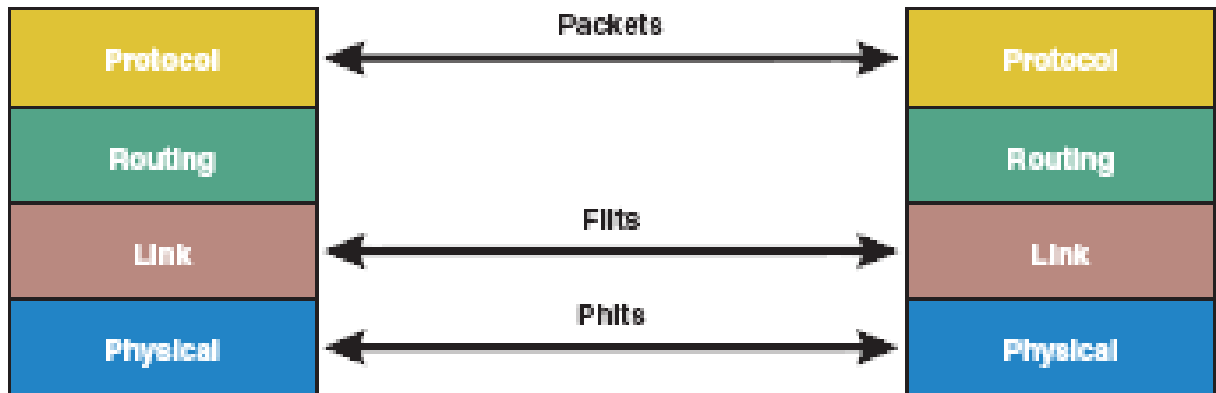
- Each **node** (which can be a multicore – remark the difference between *multinode* architectures and *multicore* chip) is connected to the system **through a high efficiency cache** and on the QPI bus it acts as a *caching agent*
- Each **node** has a private memory, that is it controls directly a portion of the global addresses which can be handled by one or more *memory controller*, each one of them is called *home agent*
- The devices which control the I/O are called *I/O agents*
- The devices which control the system *boot* are called *firmware agents*
- In each node more cores can coexist which are called *sockets*
- The interconnections (with different parallelism – see later) are called *links*
- Interrupts and power down messages too *are transmitted via QPI* that is they too are *network messages*

Quick Path



Block diagram of a *single* node with multiple cores (codenames Nehalem, Windmere, Sandy Bridge etc. – commercial names I5,I7...)

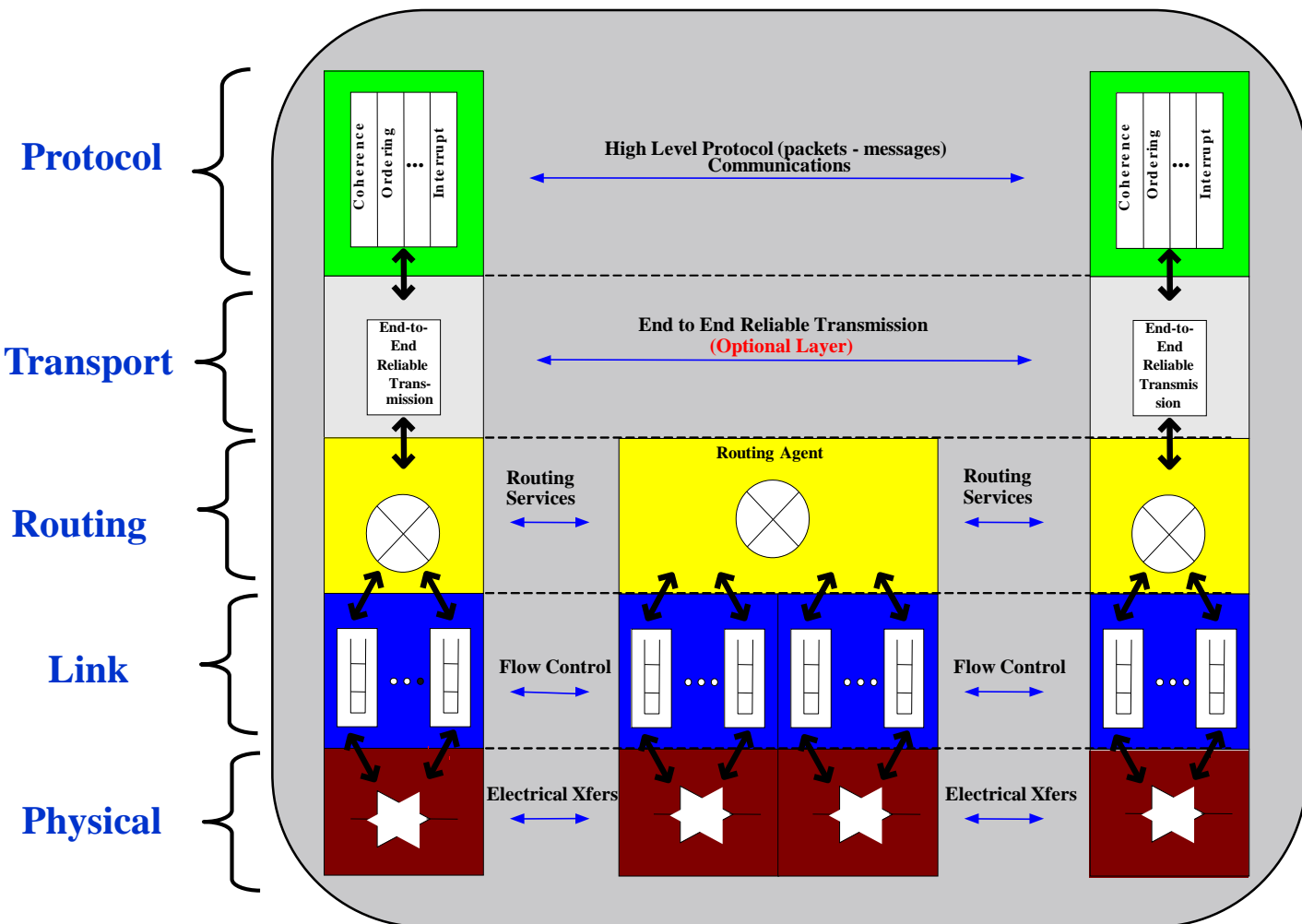
Architecture layers



- **Physical Layer:** controls the physical information exchange and the transmission errors (for example the **Cyclic Redundancy Code**). It consists of **monodirectional** connections in both directions (transmission units: **Phit of fixed length – 20 bit**)
- **Link Layer:** reconstructs the messages from the Phits and controls the information flow (messages: **Flit of fixed length – 80 bit**)
- **Routing layer:** handles the routing of messages (see – for instances – the partially interconnected previously analysed)
- **Protocol Layer:** multiple tasks. For instance an interrupt *packet (message)* can require multiple *flits* which are reassembled in the protocol layer. *Different lengths*. It implements the cache coherency protocol, handles the interrupts, the memory mapped I/O etc. More generally it handles the messages sent over multiple links which involve multiple agents

NB: The Quick Path protocol caters for the cache snoop *and allows direct cache-to-cache transfers*

Architecture layers(ISO)



Physical Layer:
High-Speed
electrical transfer

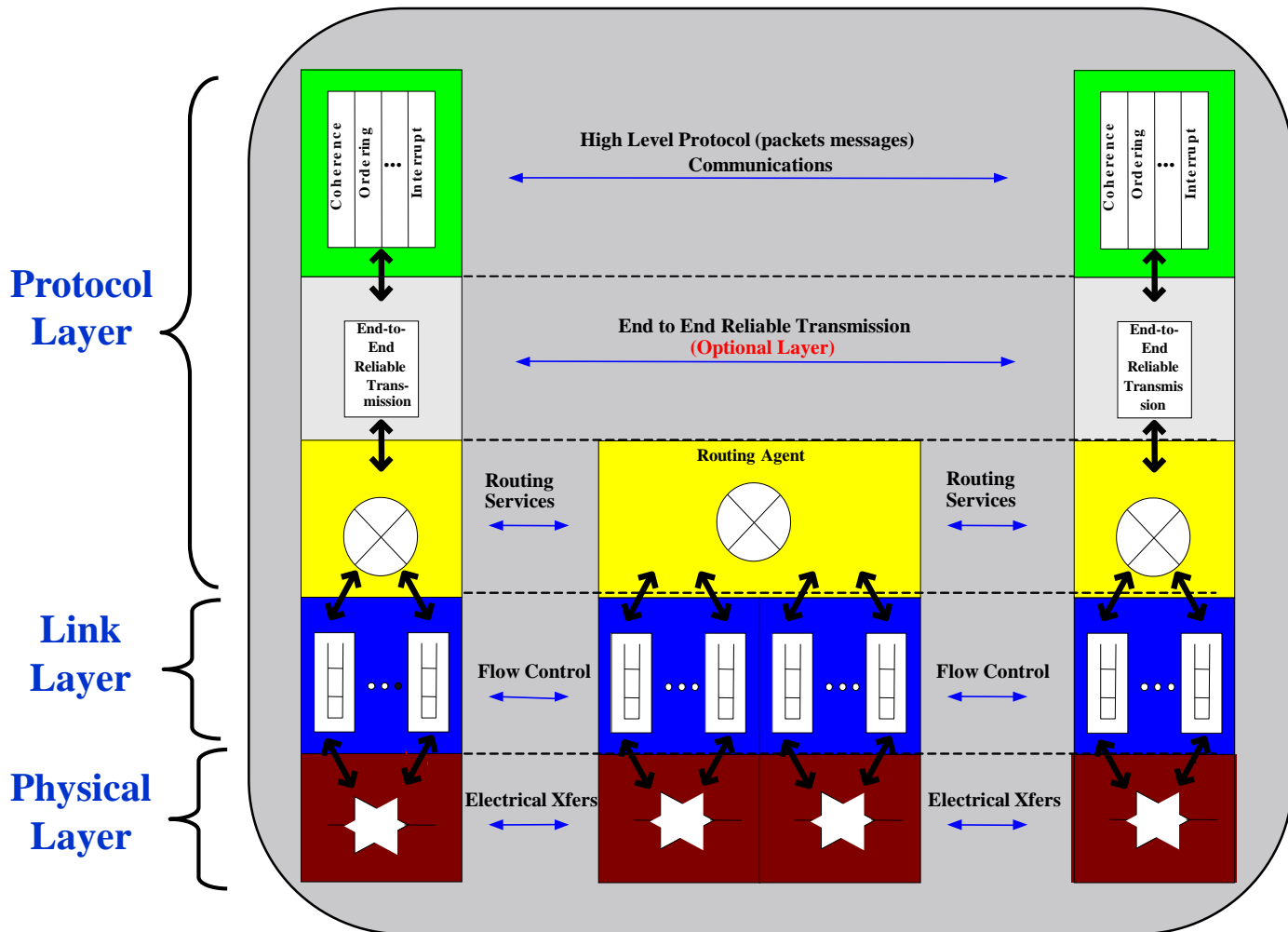
Link Layer:
Reliable
transmission, Flow
Control between
agents

Routing Layer:
Framework for
routing capability

Protocol Layer:
High Level protocol
information exchange,
High Level Commands
MEMRD, IOWR, INT
Coherency,
Packets reorder
etc

Transport Layer:
Advanced Routing
capability for
reliable end-to-
end transmission
(seldom
implemented)

Architetture layers



Physical Layer:

Operates on **PHITS** = 20 bits
 1 PHIT carries 2 BYTES of data +
 2 controls bit + 2 bits CRC
PHIT is the *minimum unit* of raw
 data

Link Layer:

Operates on **FLITS**
 1 FLIT = 4 PHITS
 1 FLIT = 80 bit
 Paylod 4x (2
 bytes/Phit)=8 bytes
FLIT is the minimum
 unit of protocol

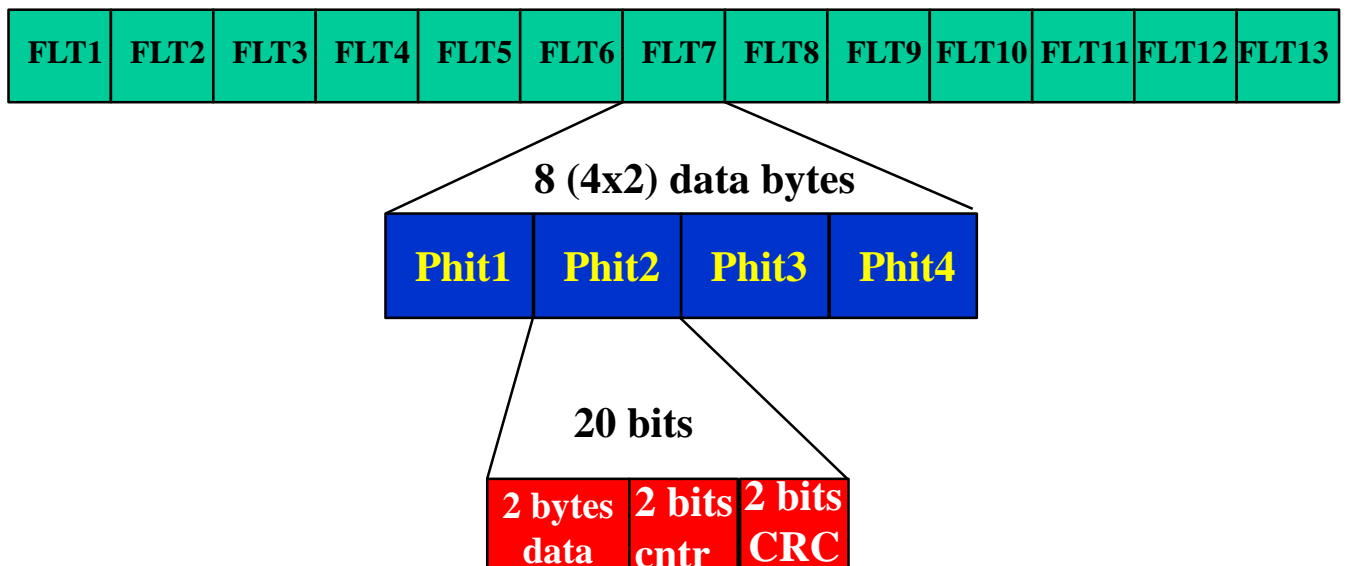
Protocol Layer:

Transmission Layer:
Routing Layer:
 Operates on **PACKETS**
 1 **Packet** = 1 or more FLITS

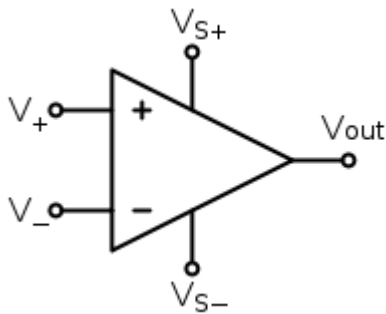
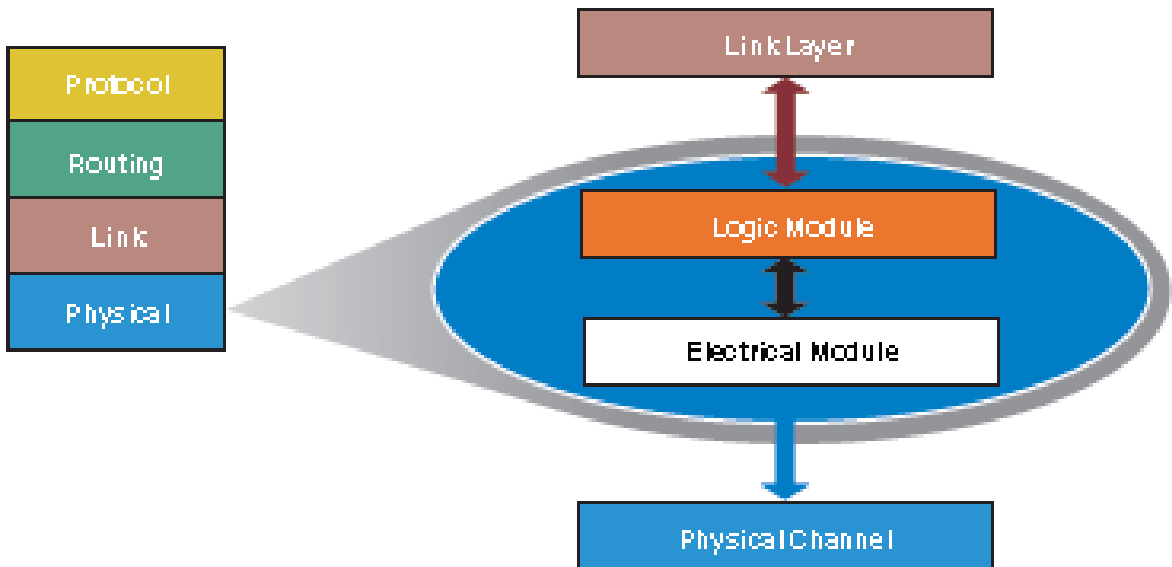
Only the protocol layer is aware of the meaning of the transmitted data

Example

This is an example of a data packet (message)



Physical Layer



Differential transmission

Transmission of a «1»

$V_+ = +0,5V$, $V_- = -0,5V$, $V_{out} = (V_+) - (V_-) = 1V$

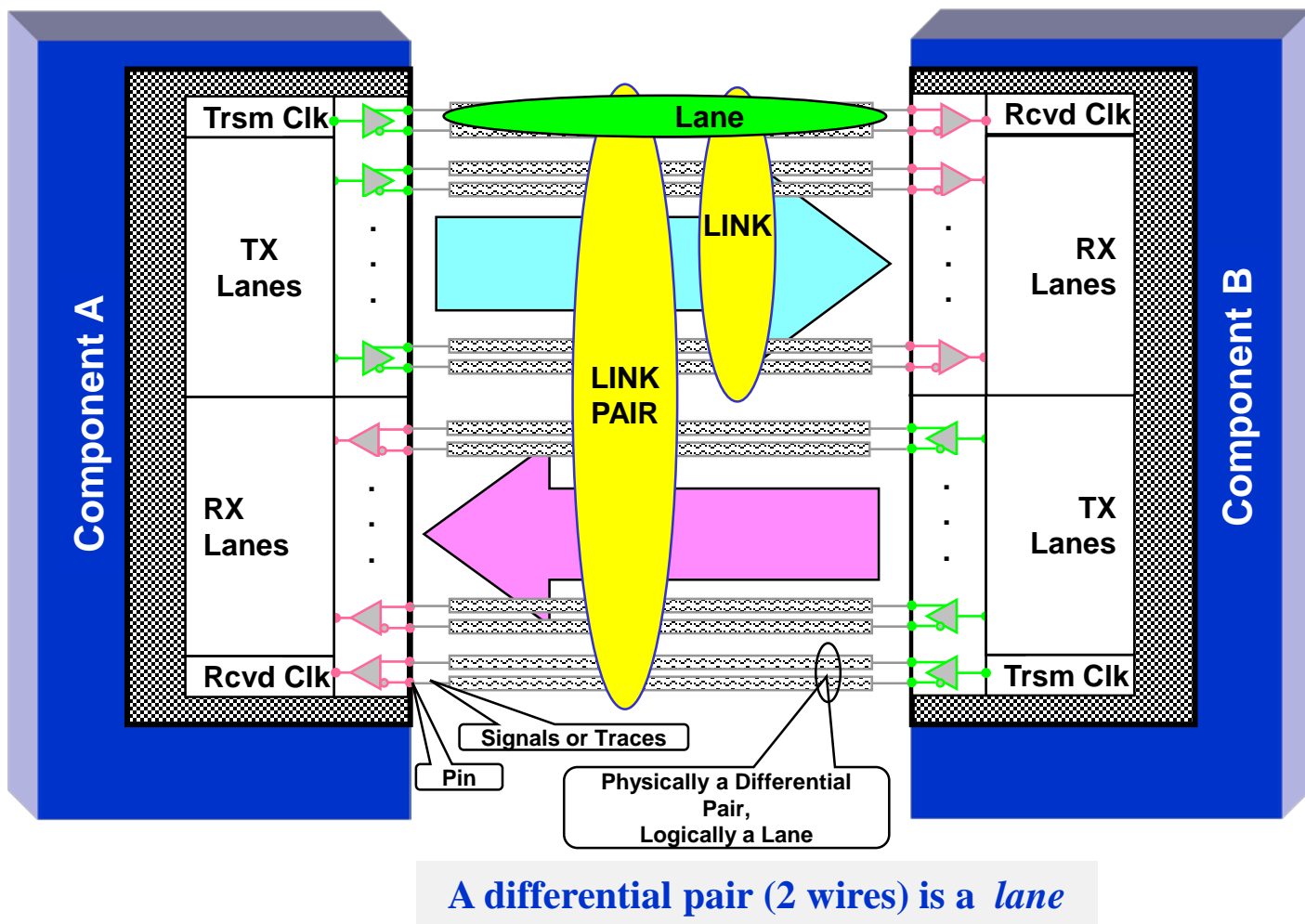
Transmission of a «0»

$V_+ = 0V$, $V_- = 0$, $V_{out} = (V_+) - (V_-) = 0V$

Smaller signal dynamic (0,5V/channel, 1V out)
Noise rejection!!!!

Voltage swing in QPI is nominally 1 V.
Maximum swing 1,36-1.38 V

Physical Layer

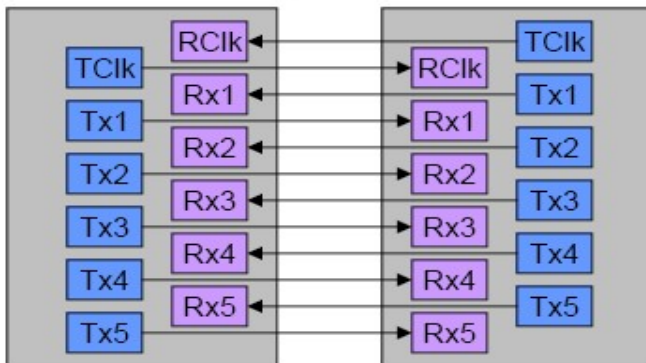


In the maximum configuration (full link) one phit each clock is transmitted in both directions

A full link has 20 data lines (1 Phit 20 [16+2+2] bit (two bytes plus 2 bit controls plus 2 bit CRC) => 40 differential wires) for each direction *plus two clocks* (one for each direction - 4 differential wires). Totally 20 data *lanes* x 2 wires x 2 directions + (2+2) clocks = 84 wires.

Physical layer

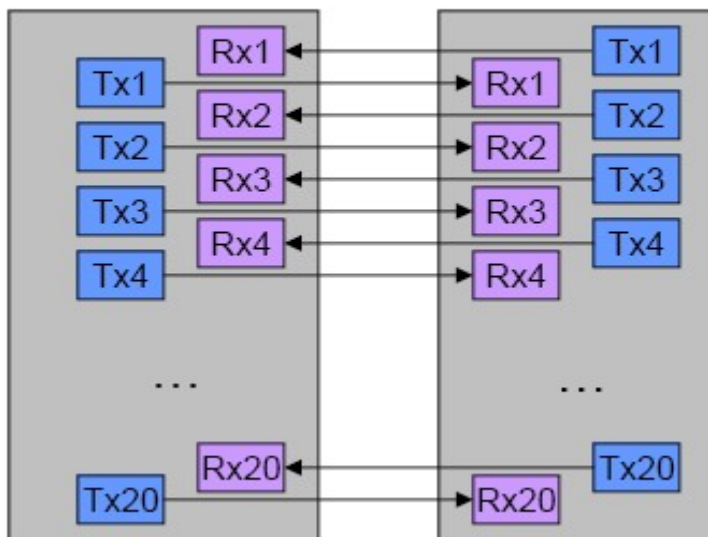
5 bit lanes and 1 clock = 1 quadrant
Differential pairs not shown



A quadrant is a reduced QPI configuration which transfers **one fourth of a PHIT** (*one Phit is 20 bits + 4 contr.*) that is it transfers 4+1 bits - differential) each clock. The single bit is in turn the control and CRC. A quadrant consists therefore of 10 data wires (5x2 differential) plus 2 wires for the clock that is 12 wires. Transmission is bidirectional so 24 wires are involved.

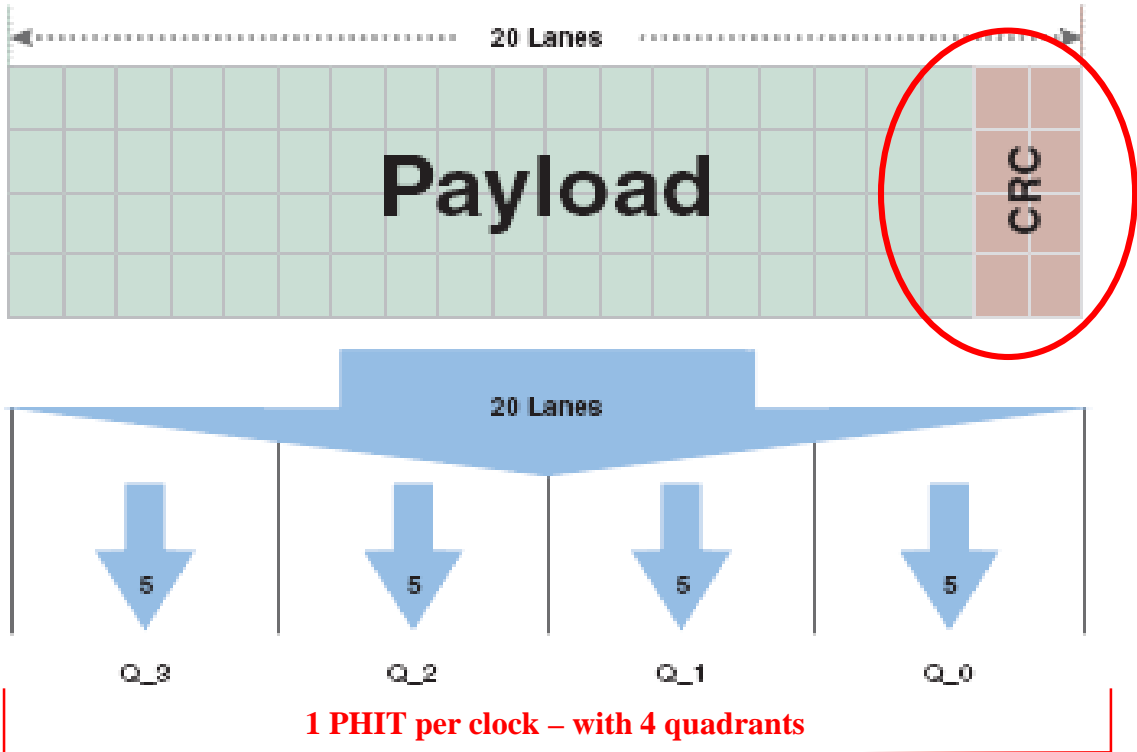
The clocks **consist always of 4 wires no matter** how many quadrants are present (a clock is common to all quadrants)

4 quadrants = 1 full link
Clocks not shown



4 quadrants (5 x 4=20 lanes) make a *full link* (20 bit transmitted - only 16 of payload – payload=> real data) which **transfers one PHIT (2 bytes + controls)** each clock. No matter how many quadrants are used (1, 2, 4) there is a single clock for each direction (in a full link there are 42 lanes - bidirectional 84 wires). CRC bits are always present

Payload



The information unit of the *link layer* is the *flit* which consists of 4 phits (and therefore is 80 bit): payload (4x2=8 bytes), control (4x2=8 bits), CRC bit (4x2=8 bit). One *Phit* is transferred each clock edge (with 20 lanes – but a physical configuration can consist of a single quadrant only - in that case more *transfers* are required for a single *Phit*).

The CRC has the following polynomial form

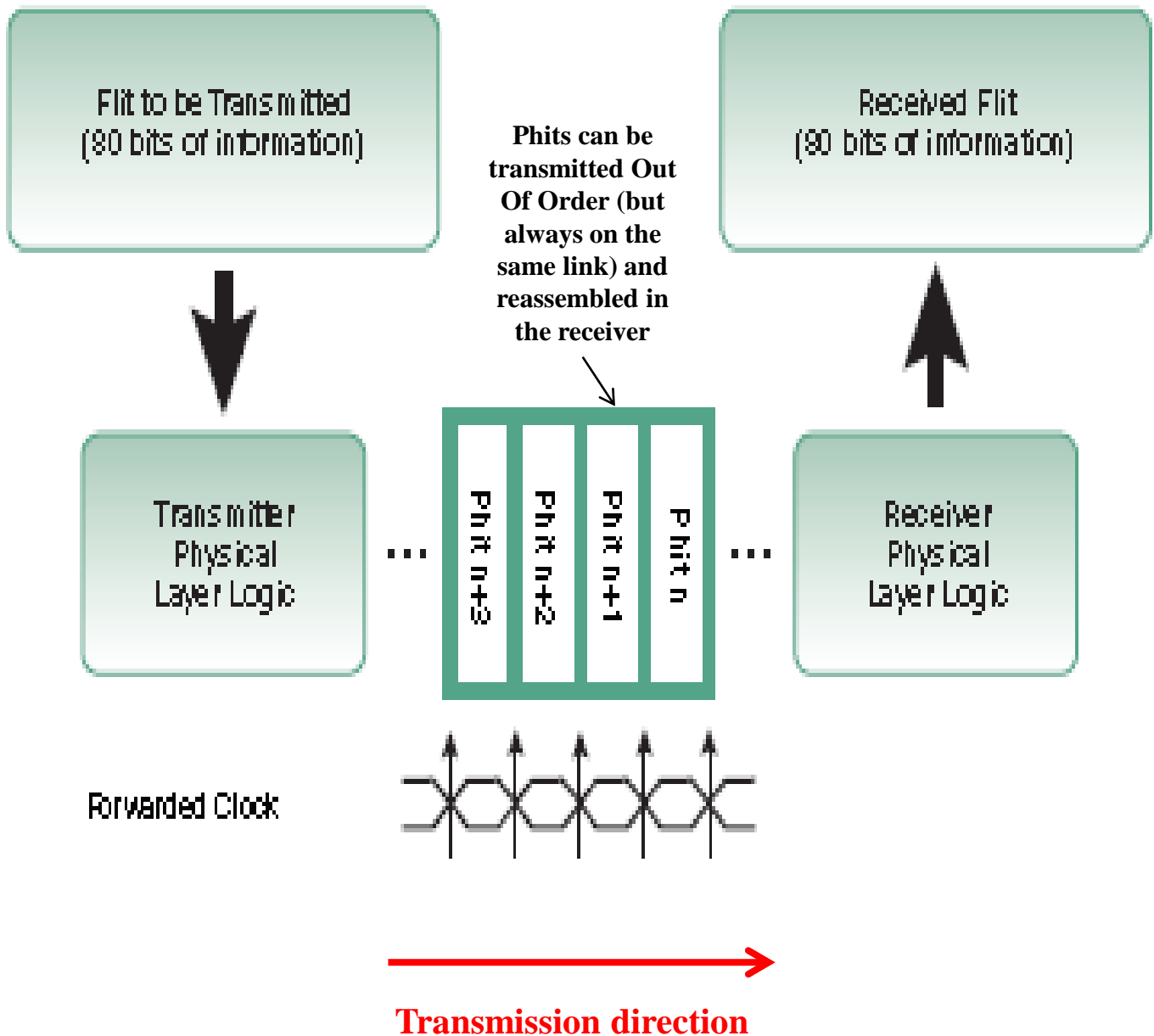
$$X^8 \oplus X^7 \oplus X^2 \oplus 1$$

which allows the correction of

- single, double and triple errors
- all errors when their numbers are odd
- any error in 8 consecutive bit
- 99% of errors in 9 consecutive bit

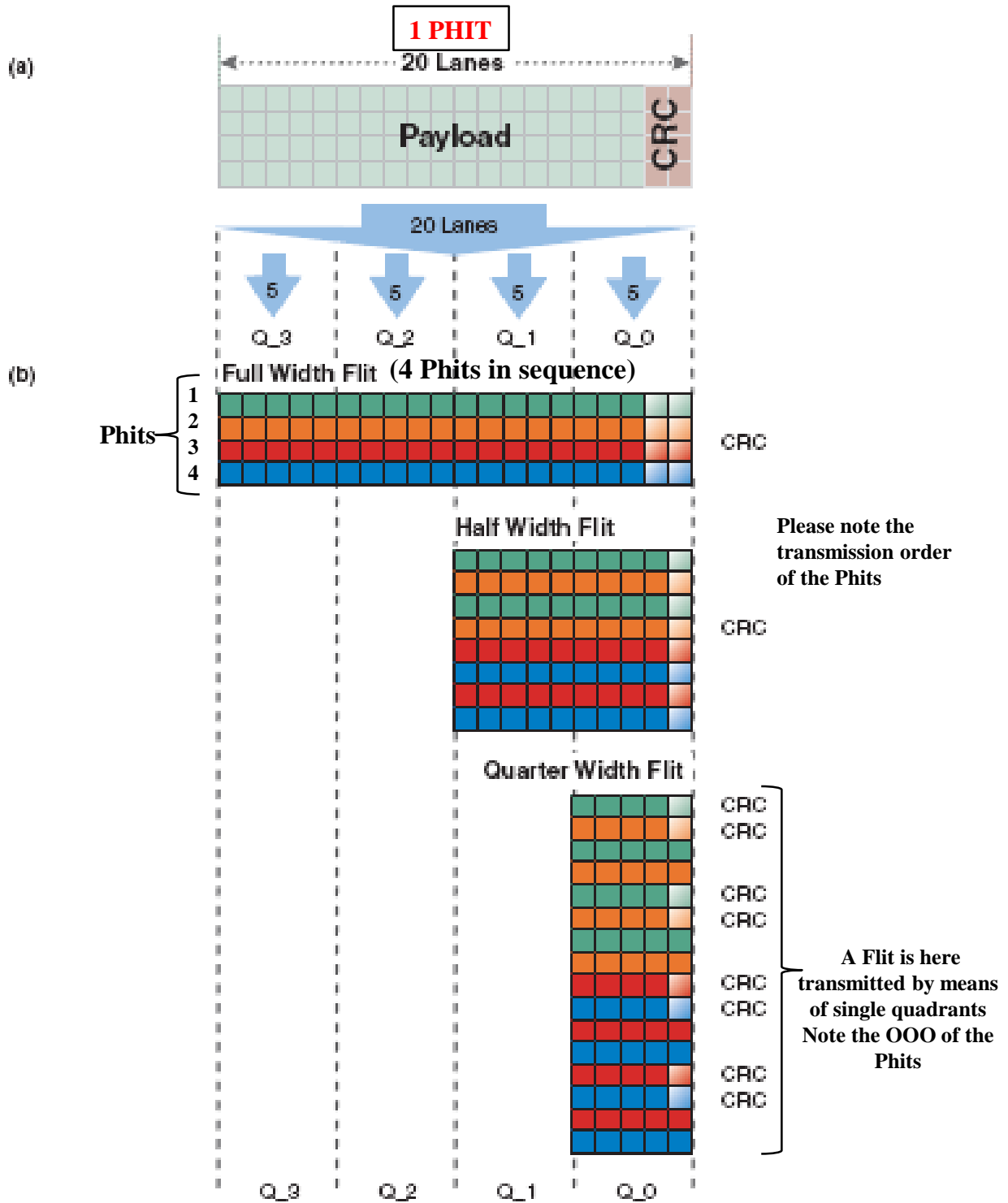
In order to grant the absolute correctness in all possible cases QPI can increase the reliability by means of an additional methodology called *Rolling CRC* which uses the CRC of the preceding Flit together to the present one leading to a 16th order polynomial.

Phit e Flit



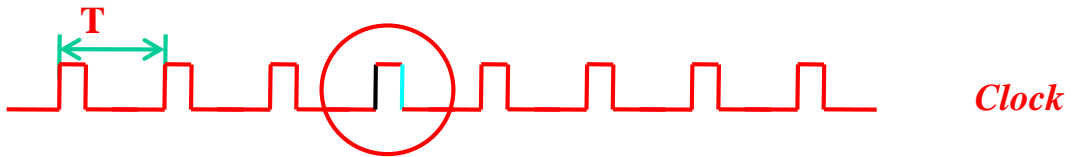
Different quadrants

Flit implication



Performance

- The theoretical bandwidth can be computed on the basis of the transmission frequency. It is the number of the transferable bytes per second
- The maximum clock frequency is (at the present time) **3.2 GHz** (twice that of the last FSB Xeon). The Quick path *transfers data on both clock edges* (**positive and negative**) and therefore there are **6.4 GT/s** (giga-transfers per second) which on 20 lanes (40 differential wires) lead to a rate of $6.4 \times 20 = 128 \text{ Gbit/sec}$ in each direction. Since the transmission is bidirectional the *maximum theoretical bandwidth* is **256 Gbit/sec**



- On the 20 lanes 16 bit only are data and therefore the *max data real rate* is **$6.4 * 2(\text{bidirectional}) \times 2(16\text{bit}/2=2\text{bytes})$ Gbytes/sec = 25.6 GBytes/sec** of information. There is one Phit (2 bytes) transfer each 156 ns ($6.4\text{GHz} \Rightarrow 156\text{ns}$) in both directions

Bandwidth (Higher is better)	Intel Front Side Bus	Intel® QuickPath Interconnect
Year	2007	2008
Rate (GT/s)	1.6	6.4
Width (bytes)	8	2
Bandwidth (GB/s)	12.8	25.6
Coherency	Yes	Yes

Performance

- What matters is the *real bandwidth* which takes into account the packets overhead.
- A typical processors transaction is the transfer of a 64 bytes cache line (**not bits** ! – that is 32 Phits – 1 Phit=16 bit=2 bytes – and therefore 8 Flits since a Flit carries 8 bytes).
- A data packet (message) requires 4 Phits for the header (8 bytes that is one Flit) plus 32 Phits (2 bytes/Phit) for the payload, that is 8 flits (total => 36 Phits => 9 flits)
- At a frequency of 6.4 GT/s (*that is a single two bytes – 1 phit - transfer each 156 psec – see previous slide*) a 64 bytes caches line requires $156\text{ps} \times 36 = 5.6\text{ ns}$.
- For the IO where packets are longer, the overhead impact is lower. In the following table a comparison with the PCI.

Overhead (Lower is better)	Intel® QuickPath Interconnect ¹	PCI Express ²
Version		Gen2
Max payload size (bytes)	64	4096
Packet payload size (bytes)	64	64
Number of packets	1	1
Payload (phits)	32	32
Header + CRC (phits)	4	7
8b/10b encoding impact ¹ (phits)	n/a	8
Total overhead (phits)	4	15
Total packet size (phits)	36	47
Total overhead (%)	11%	32%

Physical layer

	Intel Front-Side Bus ³	Intel® QuickPath Interconnect ³
Topology	Bus	Link
Signaling Technology ¹	GTL+	Diff.
Rx Data Sampling ²	SrcSync	FwdClk
Bus Width (bits)	64	20
Max Data Transfer Width (bits)	64	16
Requires Side-band Signals	Yes	No
Total Number of Pins	150	84
Clocks Per Bus	1	1
Bi-directional Bus	Yes	No
Coupling	DC	DC
Requires 8/10-bit encoding	No	No

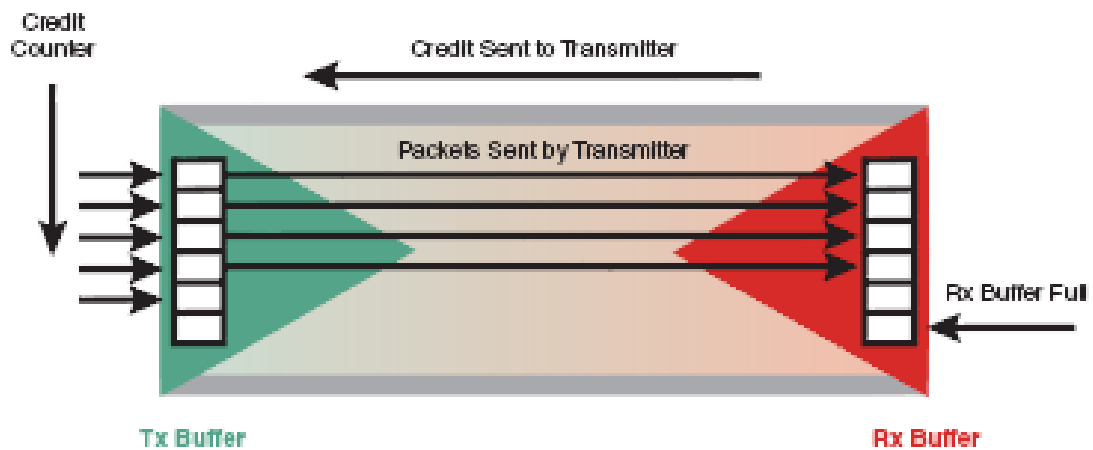
Voltage swing: 1 V

Physical layer

- A QPI link is a point-to-point interface of variable size, based on packets, which consists of bidirectional differential lines
- The clocking mechanism has no particular features which in the future makes possible *optical implementations*
- In order to detect the max allowed transfer frequency and cater for the changes due to temperature, voltage etc. variations, both at the startup and during the normal behaviour calibration tests are carried out by means of specific circuits (*normally off*) . During these tests the high level functions are temporarily suspended
- Some of the data lanes can be converted into clock lanes in case of clock lane failure (reduced efficiency but not blocked system behaviour)
- Lanes directions can be inverted
- The physical layer carries out periodical tests in order to check the Bit Error Rate. Among the tests: the loop back and the Interconnect Built-In-Test which allows to test the link at the maximum speed

Credit/debit scheme

- For the flow control the link layer uses a credit/debit scheme



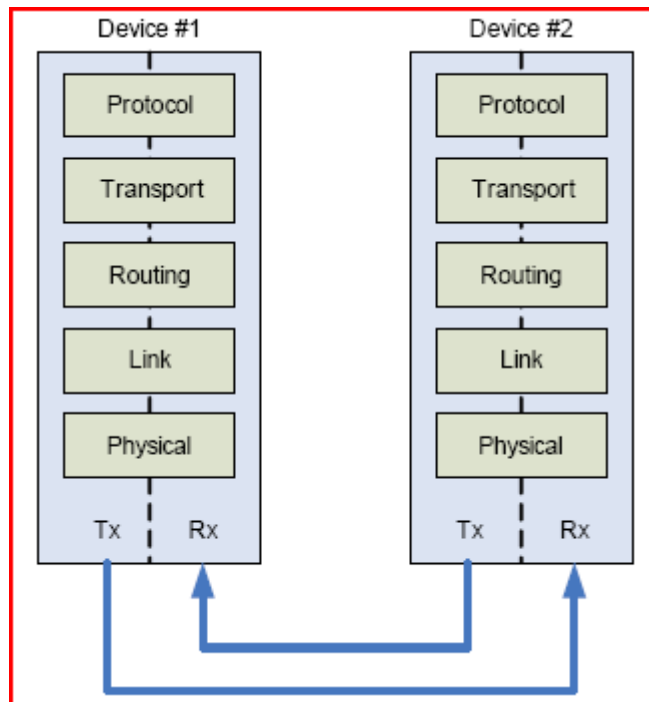
- At the startup the **transmitter** receives from the receiver a number of “credits” for the Flits transmission decremented for each completed transfer
- When the receiver buffer is emptied a further credit is sent to the transmitter
- If the transmitter has no more credits it stops transmitting
- Control informations are carried piggyback

Permanent errors solution

Feature	Intel® QuickPath Interconnect
CRC checking	Required
All signals covered (by the CRC)	Yes
CRC type	8 bits / 80 bits
CRC bits/64-B packet	72
Impact of error	Recovered
Cycle penalty ¹ (bit times)	None
Additional features	Rolling CRC
¹ Lower is better. The cycle penalty increases latency and reduces bandwidth utilization.	

- **Reduced Services in case of high failure rate (autorepaired). Possible workarounds**
 - **Parallelism reduction (half or a quarter – 10 or 5 bit for a 20 bit link – 2 and 1 quadrants – selecting the quadrants which operate properly)**
 - **In case of clock failure, the clock is redirected onto a data lane (reduced transfer rate obviously)**
 - **NB: a link can operate at reduced rate in one direction and at full rate in the opposite direction**
 - **All this solutions are carried out without *software intervention* which is however informed about**

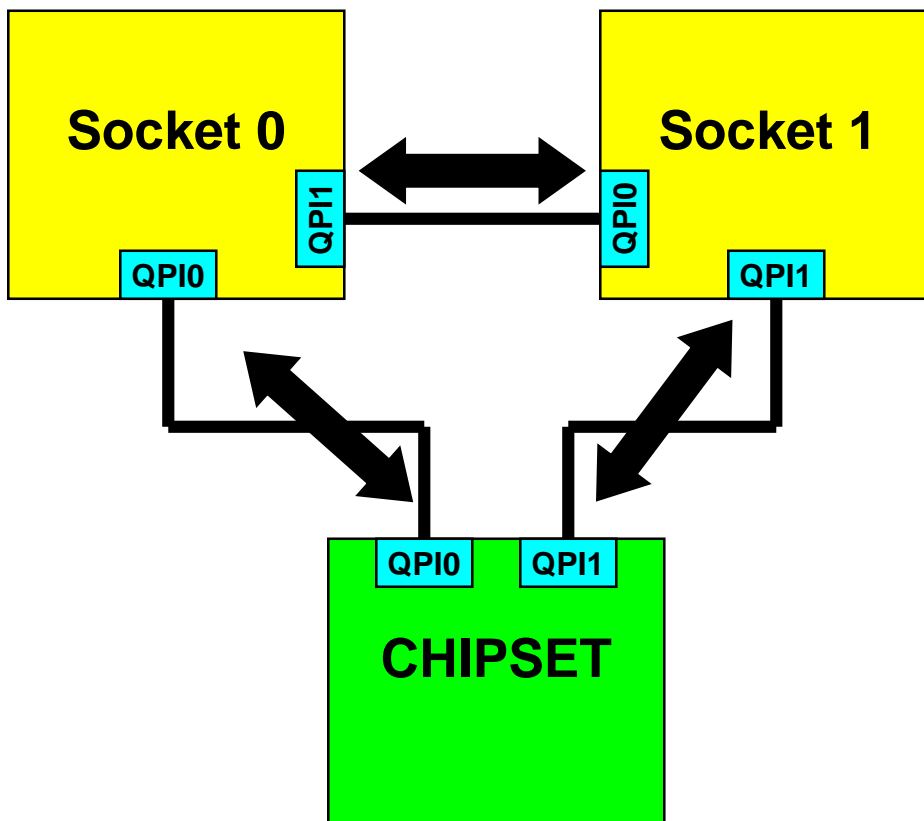
Quick Path



The protocol structure conforms to the ISO_OSI scheme

- **The physical layer consist of monodirectional connections in both directions.**
- **The link layer is responsible for the transmission correctness and flow control. No re-routing for link layer**
- **The routing layer choses the packets path in multihops systems. It maintains the routing tables for reaching all destinations**
- **The transport layer (often not implemented) provides an advanced routing capability for a reliable end-to-end transmission**
- **The protocol layer consists of rules for the coordination of the *caching and home agents (see later)***

Nodes identification

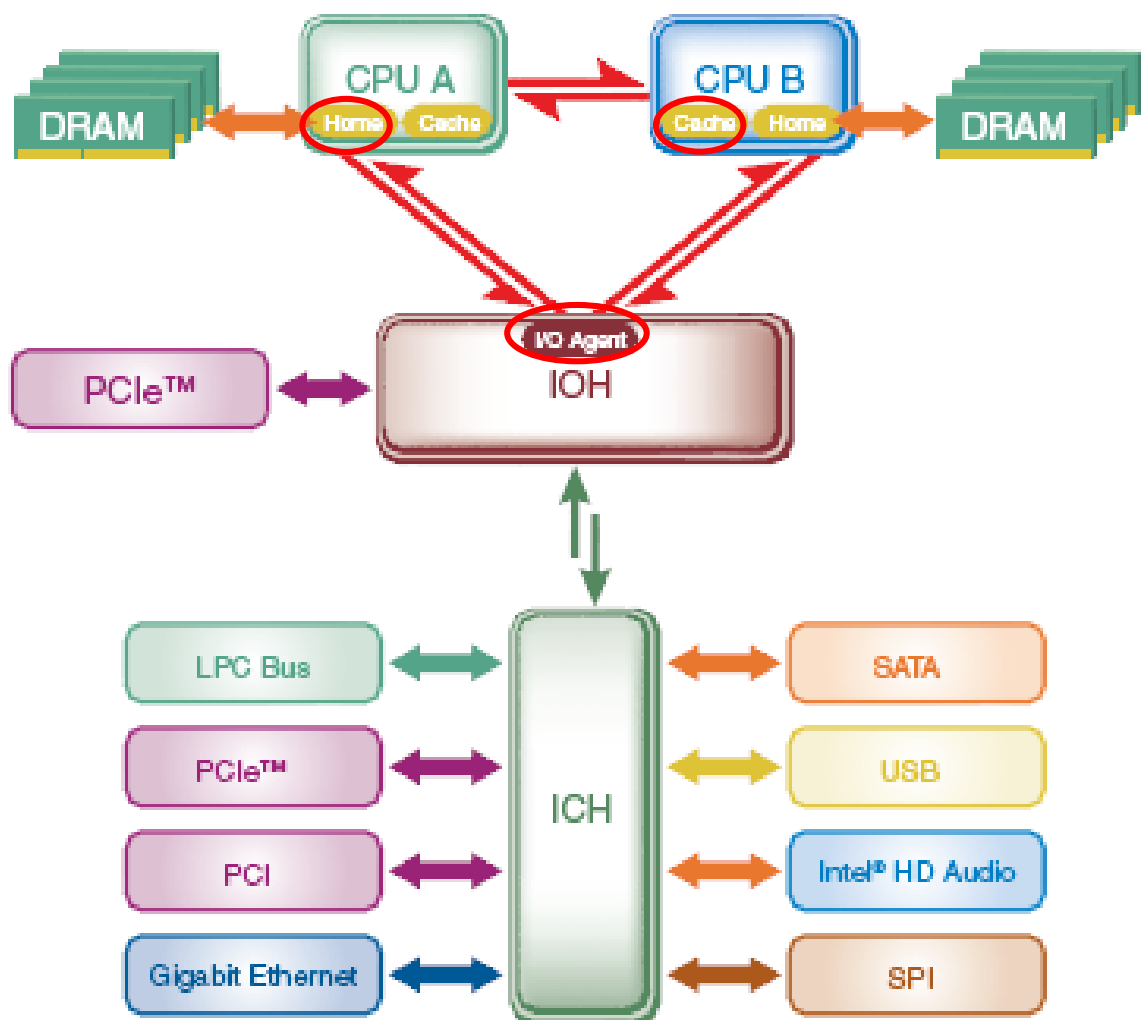



Nodes identification, agents identification, power management etc. at the startup

Each node (socket) can have multiple QI interfaces

Routing

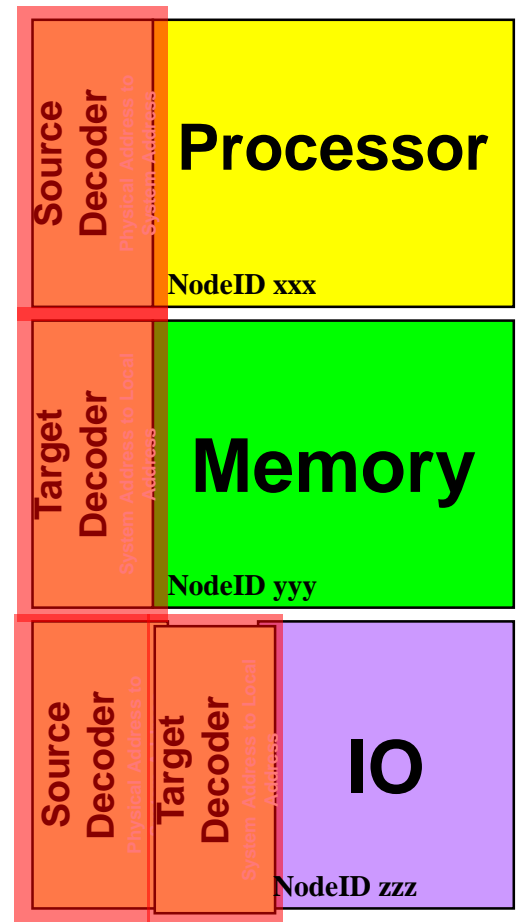
- At the start-up or Reset an unique *identifier* is assigned to each agent. In figure a biprocessor case



 => Agents

QPI addressing

- **Virtual address:** processor applications and drivers address
- **Physical address :** the address after the conversion virtual->physical
- **QPI address:** system address consisting of the *physical address and the NodeID* which points to a single QPI device in the system
- **Source decoder:** it generates the requests
- **Target decoder:** it answers to the requests



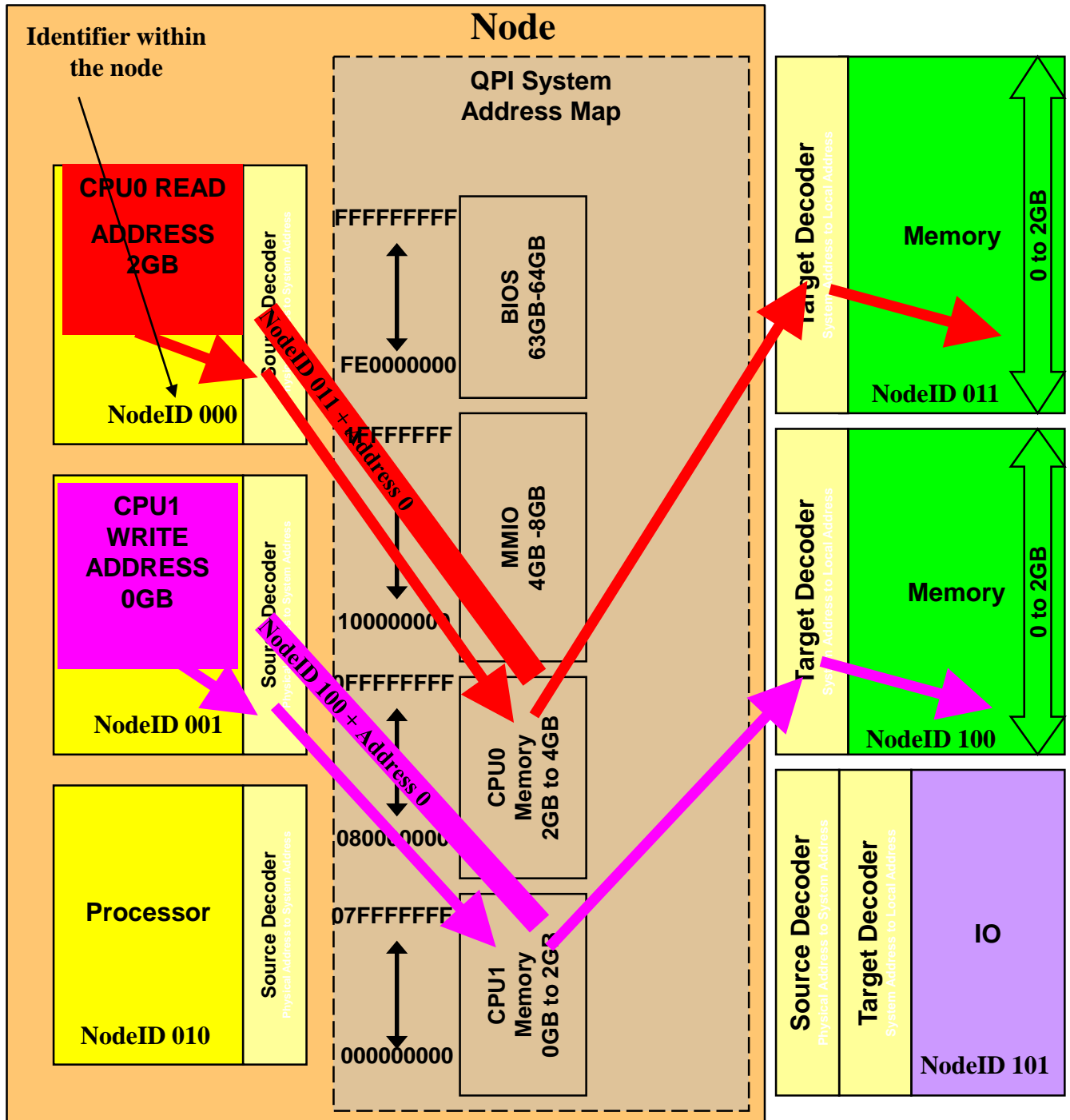
Source Decoder Example Physical Address to System Address

Source NodeID 001

CPU Physical Address		QPI System Address
Memory 00000000-7FFFFFFF	Maps To	NodeID 010 + 00000000-7FFFFFFF
Memory 80000000-DFFFFFFF	Maps To	NodeID 011 + 00000000-5FFFFFFF
Memory E0000000-FFFFFFF	Maps To	NodeID 101 + 00000000-1FFFFFFF
I/O 0000-0FFF	Maps To	NodeID 101 + 0000-0FFF
I/O 1000-FFFF	Maps To	NodeID 110 + 0000-EFFF

Hex D-8=5

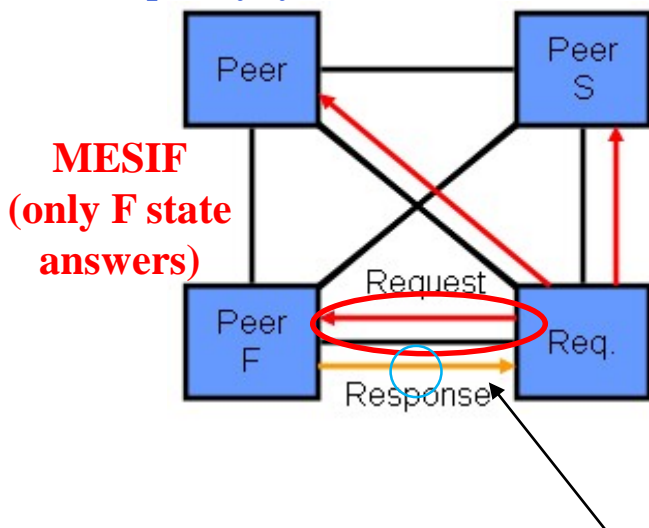
QPI address logical mapping



- The routing from the source to the destination is determined by the decoders of each QPI agent
- Each QPI agent is uniquely identified by its type and the node identifier

MESIF

- In addition to the «classical» MESI states a further state is introduced: **Forward** which is a variant of the S-state (shared). Normally the F-state is given to the agent which last received a shared line
- Only one agent can have a line in F-state: the others (if any) have the same line in S-state
- A line in M-state (modified) has nothing to do with F-state which is related *only* to S-state
- When a line which is in F-state is copied into another agent, the F-state is transferred to the new agent and the previous F-state is converted to S-state.
- When there is a broadcasted request for a line *only the agent which has the line in F-state responds* reducing the overall bandwidth occupation
- The MESIF in some cases can be not implemented (for instance in case of low complexity systems)



Broadcast to all agents BUT ONLY the cache in F-state answers

Coherency agents

- A **coherency agent** is a QPI agent which caters for the caches coherency
- There are two types of coherency agents: caching agents and home agents (directory based coherence)

Caching agents (hw)

- They handle the read and write requests in the coherent memory space
- They provide copies of the cache lines to other agents

Home agents (hw)

- They control a portion of the coherent memory space
- They record the caches state transitions
- They are interfaced with the local memory controllers
- They provide the data responses and/or «ownership» on request

QPI snoop methodology

Read

- **Source snooping:** small systems. Cache agent A broadcasts a line read request to all agents and to the *memory home agent* Q which is the owner of the line in its memory. If any cache agent has the line (for instance B) in **modified-state** (that is the line is different from the memory) it sends the line both to A and Q (which writes the line in the memory). The state line in B becomes **shared** and in A becomes **forward** (if implemented). If the line in B was in **forward state** B sends the line to A and its state becomes **shared** and in A the state is **forward** (if implemented). If no cache stores the line, the line is provided by Q *memory home agent* and the state in A is **forward** (if implemented). If F state is not implemented the line in A in this case is *shared*. In any case the line arrives to A *in two steps* (request and transfer). Maximum transfer speed
- **Home snooping:** (large systems). In this case A requests the line to the *memory home agent* Q which maintains a list of all agents having the line in their caches (directory based system). Q broadcasts the request to all agents having the line: data is provided with the previous rule. In this case three steps are required. A->Q, Q->B, B->A where B is the agent which has the line in forward or modified state. The advantage of the home snooping is the reduction of bandwidth occupation (***broadcast only to selected agents***) but reduced efficiency (three steps). Minimum bandwidth occupation

QPI snoop methodology

Write

- **Source snooping:** small systems. Agent A broadcasts a line write request to all agents and to the *memory home agent* Q which is the owner of the line in its memory. If any agent has the line (for instance B) in **modified-state** it sends the line to A and invalidates the line. The line is then written by A locally where it becomes **modified**. If the line was in **shared or forward** state the line is sent to A (either by the cache where it is in **F-state** or by *memory home agent* Q if F not implemented – minimum bandwidth occupation) and then all caches (if any) invalidate the line. In any case the line arrives to A *in two steps*. An invalidation of a modified line in a cache triggers a write-back to Q. Maximum transfer speed
- **Home snooping:** (large systems). In this case A requests the line to the *memory home agent* Q which maintains a list of all agents having the line in their caches (directory based system). Q broadcasts the request to all agents having the line: data is provided with the previous rule. In this case three steps are required. A->Q, Q->B, B->A where B is the agent which has the line in forward or modified state. The advantage of the home snooping is the reduction of bandwidth occupation (**broadcast only to selected agents**) but reduced efficiency (three steps). Minimum bandwidth occupation

Addressing space

- **Non-coherent memory:** addresses which do not follow the coherency rules and therefore cannot be **cached**
- **Memory mapped I/O:** *obviously* non-coherent
- **I/O:** not coherent
- **Interrupt and other special operations:** these are the addresses used – for instance – for the interprocessor interrupts

A two processors message example

Coherent memory read

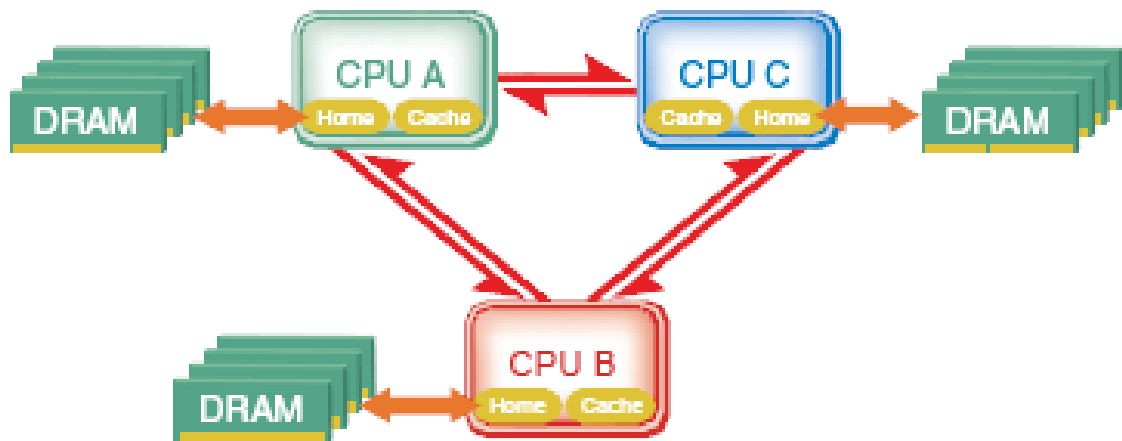
Read source snoop in a biprocessor (two nodes)

- 1) Processor 1 cache agent sends a **snoop** message for a line to the processor 2 *caching agent* and a **read** message to the processor 2 *home agent* (this is simple biprocessor system for the example)
- 2) Processor 2 snoop agent can respond:
 - a) **Invalid**. In this case the cache agent informs by a message its home agent of its **invalid state**. The line is sent to the processor 1 caching agent *by the processor 2 home agent*.
 - b) **Modified**. Two messages. The first one to the processor 1 caching agent with the line and the information that it has transformed its line state to **shared**. The second to its home agent about line and the new state and the line is updated in **memory too**
 - c) **Forward (shared)**. In this case (the line in cache is identical in memory) the processor 2 caching agent sends the line to the processor 1 caching agent and its state becomes **shared**.

Notice that the line state in cache 2 can't be **shared** in this case (biprocessor) otherwise the line would be already in the cache of processor 1!

- 3) The home agent of processor 2 acts differently according to the response of its caching agent :
 - a) **Invalid**. The home agent sends the line to the processor 1 caching agent (which becomes forward – or shared if forward not implemented)
 - b) **Forward (shared)** The home agent is aware that the line is sent by its caching agent and sends only a «concluded» message to the processor 1 caching agent

Triprocessor



Messages in a tri-processor

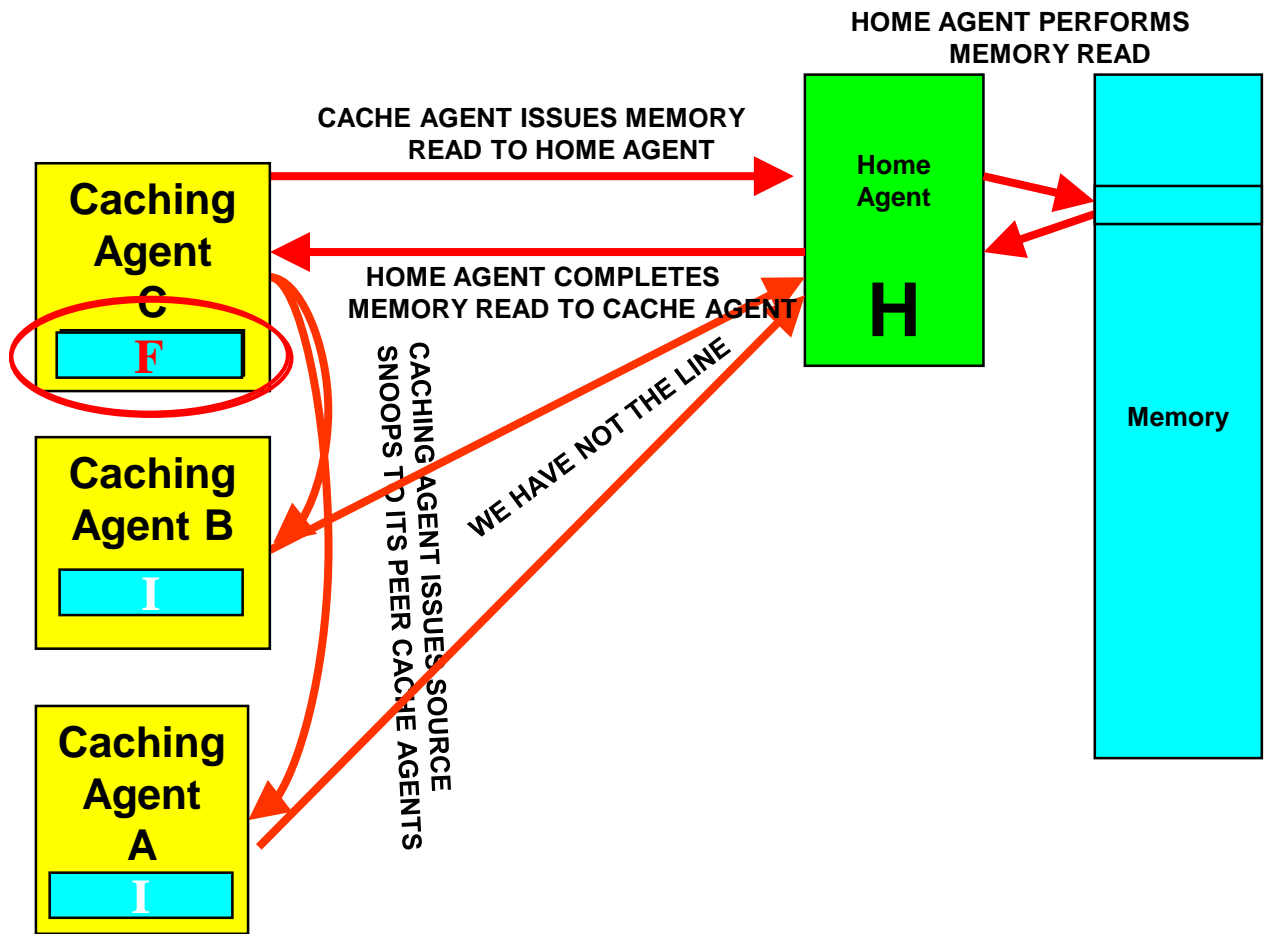
ABC	Caching Agents
H	Home Agent
MC	Memory controller

This next slides must be viewed using its .PPSM version

Snoop source of a non cached line

«Clean» line read

(Forward implemented)

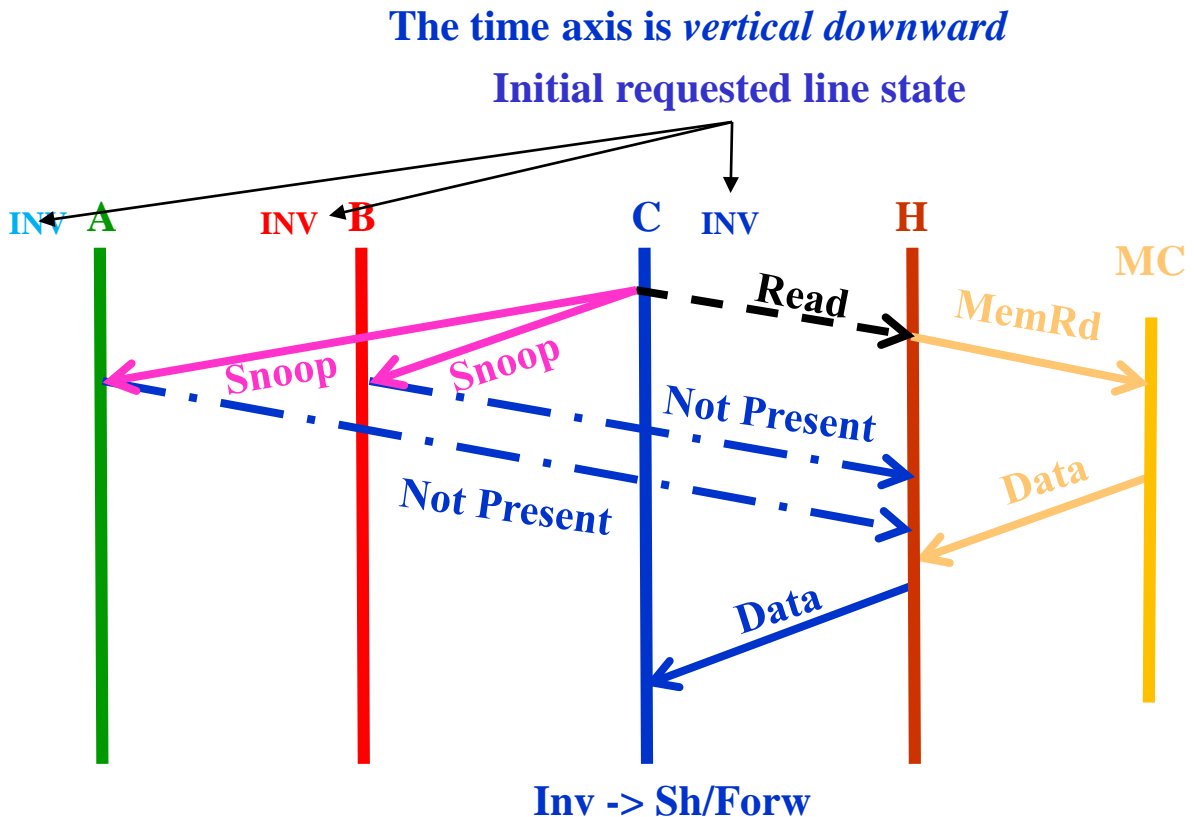


The caching agent of processor (C) snoops all other caches. If the line is *not present* in A and B the *home agent sends the line*

Snoop source read of a non cached line

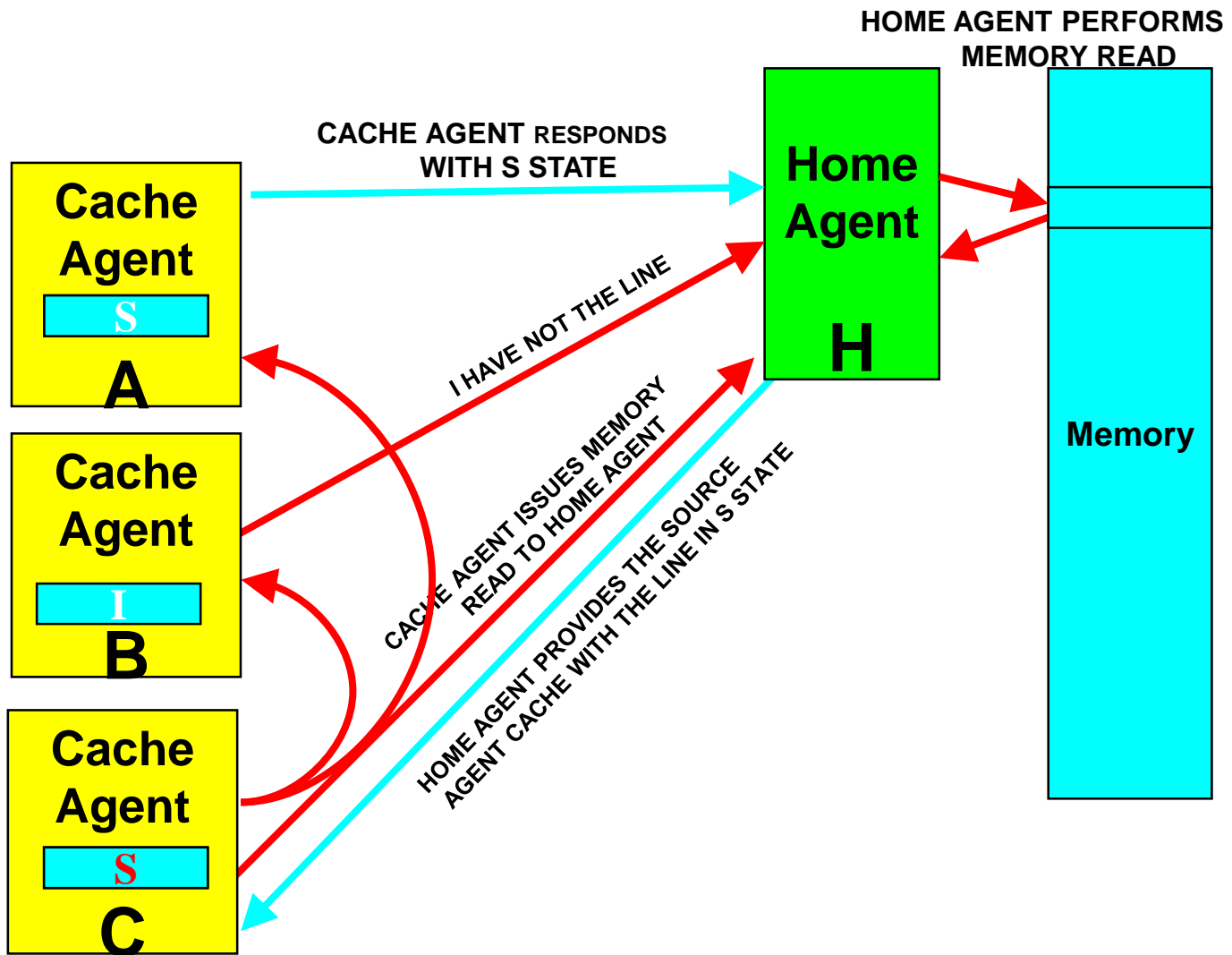
«Clean» line read

(Forward implemented)



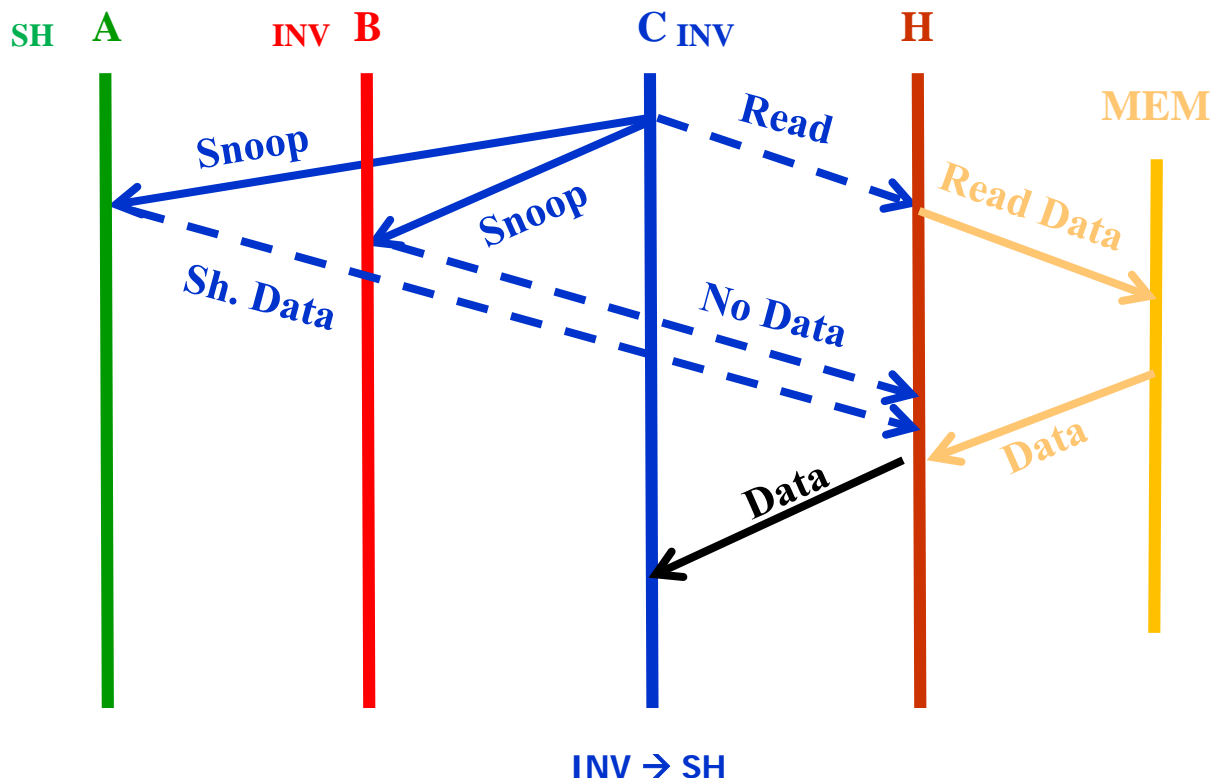
- 1) The caching agent C requests a line. It snoops the caching agents A and B and requests the line to the home agent of the line too.
- 2) The home agent waits for the response of the caching agents A and B which haven't the line (Invalid). The home agent reads the line from its memory and sends it to caching C indicating that the line must be in Forward (F) state

Snoop source read of a cached line (no Forward implemented)



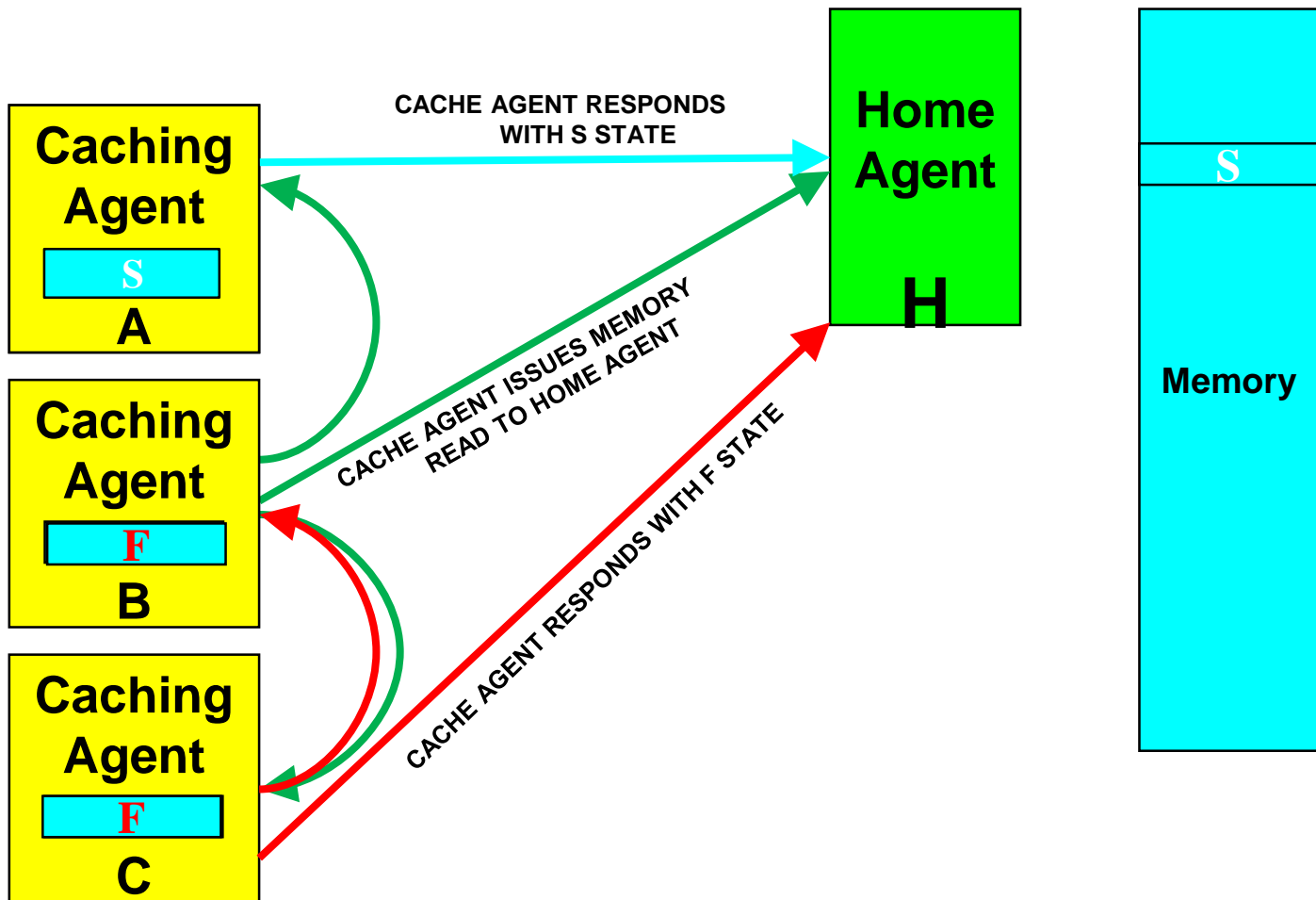
After C snoop (no cache has the line in F state – i.e F state not implemented).

Snoop source read of a cached line (no Forward implemented)



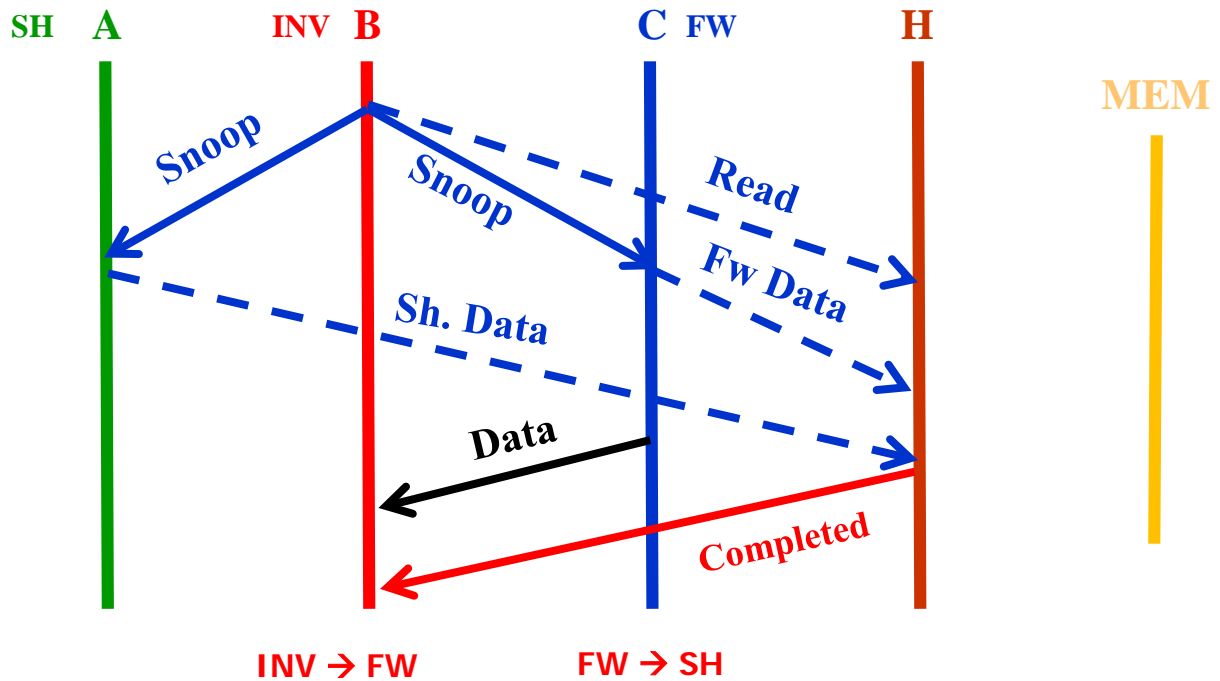
In this figure (three processors) C requests a line not present in its cache (invalid). The line IN a is in S state. After the snooping the line in A is still in S and in C in state S (as requested by the home agent). *It must be noticed that in this case too the line is provided by the home agent (a cache with a line in S state does not provide the line because it doesn't know whether there is another line in S state)*

Snoop source read of a cached line (Forward implemented)



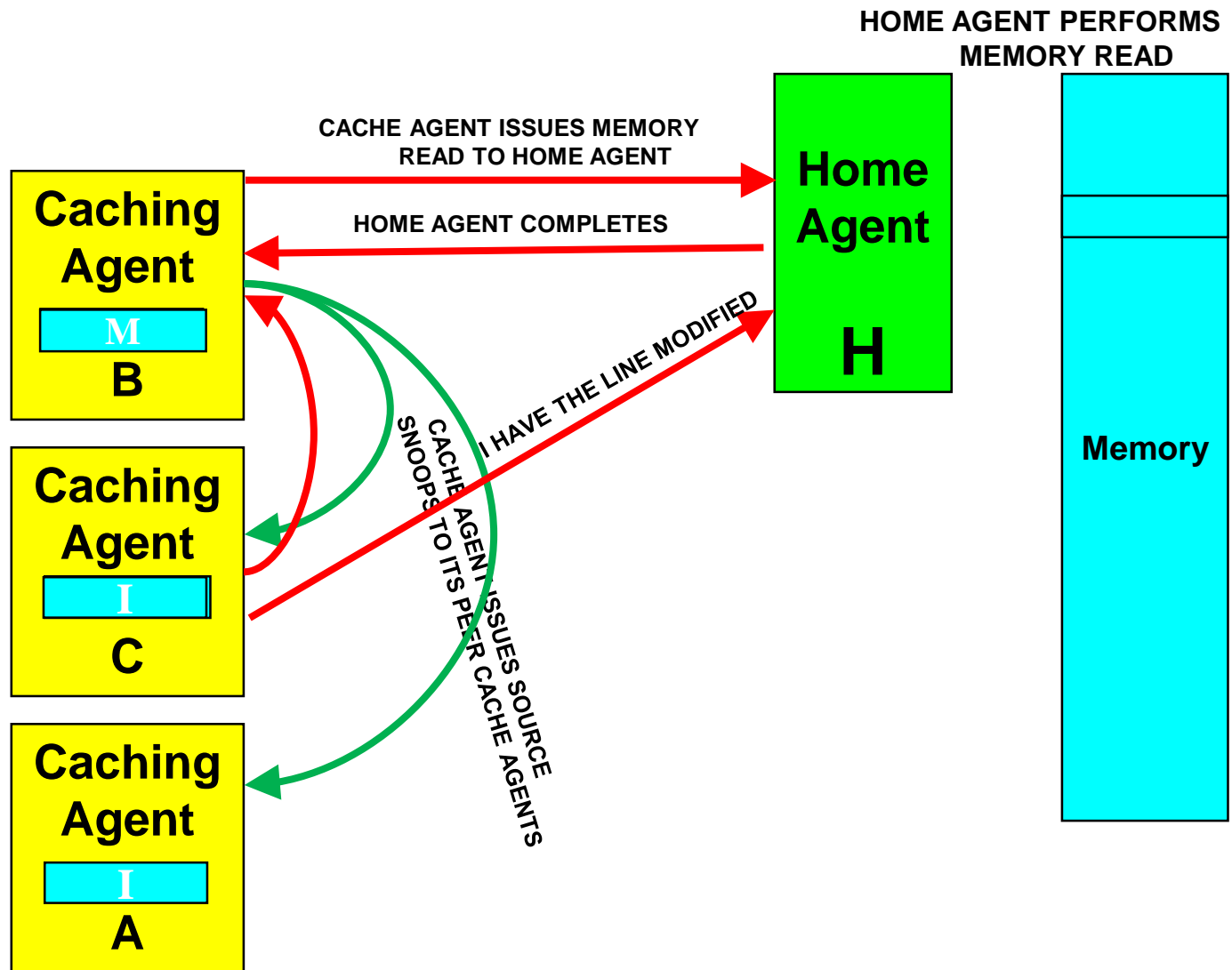
- The line in F state in C is sent to the requesting B *without requesting* a copy to the home agent. The line in B eventually becomes F
- If no node has the line in F state what agent should provide the line? Previous case: the home agent

Snoop source read of a cached line (Forward implemented)



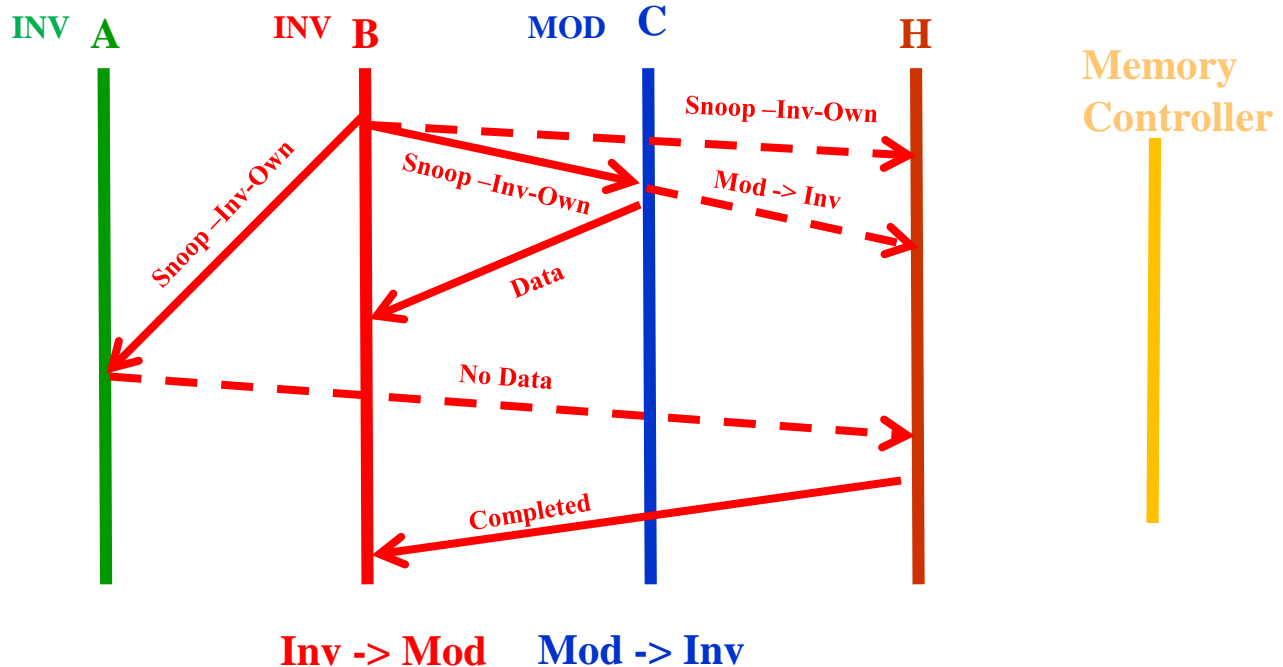
In this figure B requests a line invalid in its cache (or not present- it is the same) which is present in C in F state and in A in S state. In this case the line is provided by C; in C the state line becomes S while in B becomes F.

Snoop source write of a cached «modified» line



A requests the line. Cache-to-cache line transfer

Snoop source write of a cached «modified» line



- 1) B wants to write a cache line. The line in B can be invalid, shared, modified or exclusive state.
- 2) If the line in B is in invalid a request is sent to the line home agent (for the case the line is not in any cache). All other caching agents are snooped for exclusive ownership of the line. If the line in B is shared only snoop of other caches
- 3) Three possible responses from the other caching agents:
 - *Invalid*. Message to the home agent
 - *Shared (or Forward)*. state transformed in *invalid* and a message to the home agent
 - *Modified* (as in figure for C – in B the line is invalid). The line is sent to B by C indicating that its state is *modified* and message is sent to the home agent signalling that the line state in C is now *invalid*.
- 4) The home agent sends the line to B when no caching agent sends the line. In any case the line is NOT written back to the memory
- 5) In B the line state becomes *modified*

Home snoop

- It is a *directory based* coherency methodology (traffic reduction but reduced efficiency)
- Four steps protocol
 - Step 1: the caching agent requests the line to its home agent
 - Step 2: the home agent snoops the caching agents which could store a copy of the line according to a line directory
 - Step3: the caching agent “snooped” sends the line state and the line (if any) to the home agent and to the requesting node
 - Step 4: the home agent provides the line to the requesting agent *if no other agent has provided it*
- A good methodology for very complex systems with many agents

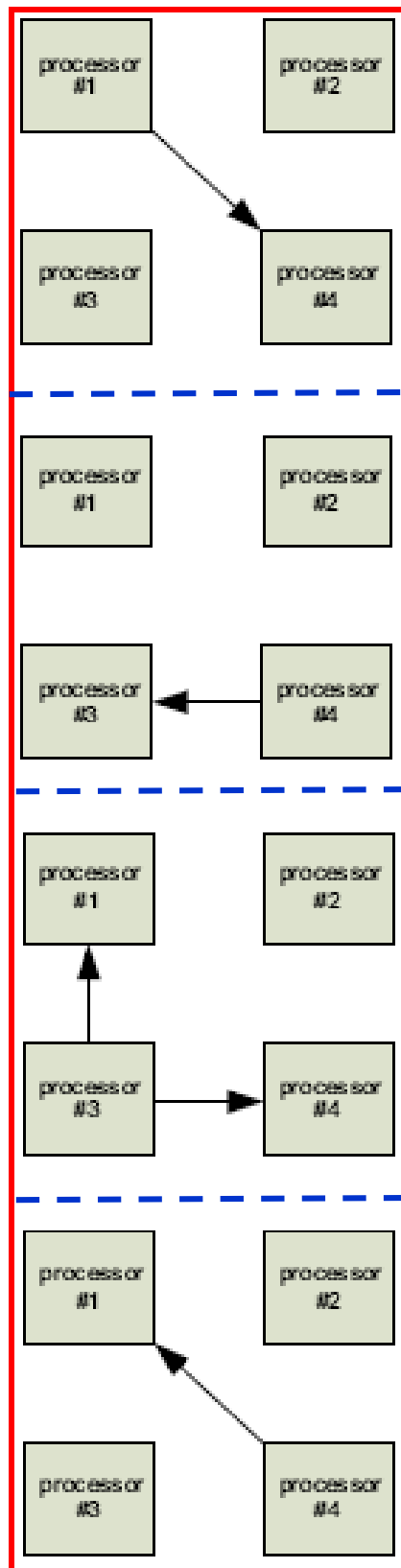
Home snoop read example

Step 1: P1 wants to **read** a line of home agent P4

Step 2 : P4 (home agent) checks its directory and send a requests to P3 **only** - P3 state (F, S or M)

Step 3: P3 response to P4: *I have the line*. The line is sent to P1 by P3. P4 (home agent) takes notices that the line in P3 is now S and in P1 in F

Step 4: P4 ends the transaction



P1 is the requesting caching agent

P4 is the home agent of the line which knows where the line is present (in many processors if the line is F or S, in a processor if M)

In this example we suppose that P3 has a copy of the line with state M or F