1394
HCI
OPEN

Implementation Examples

Peter Teng

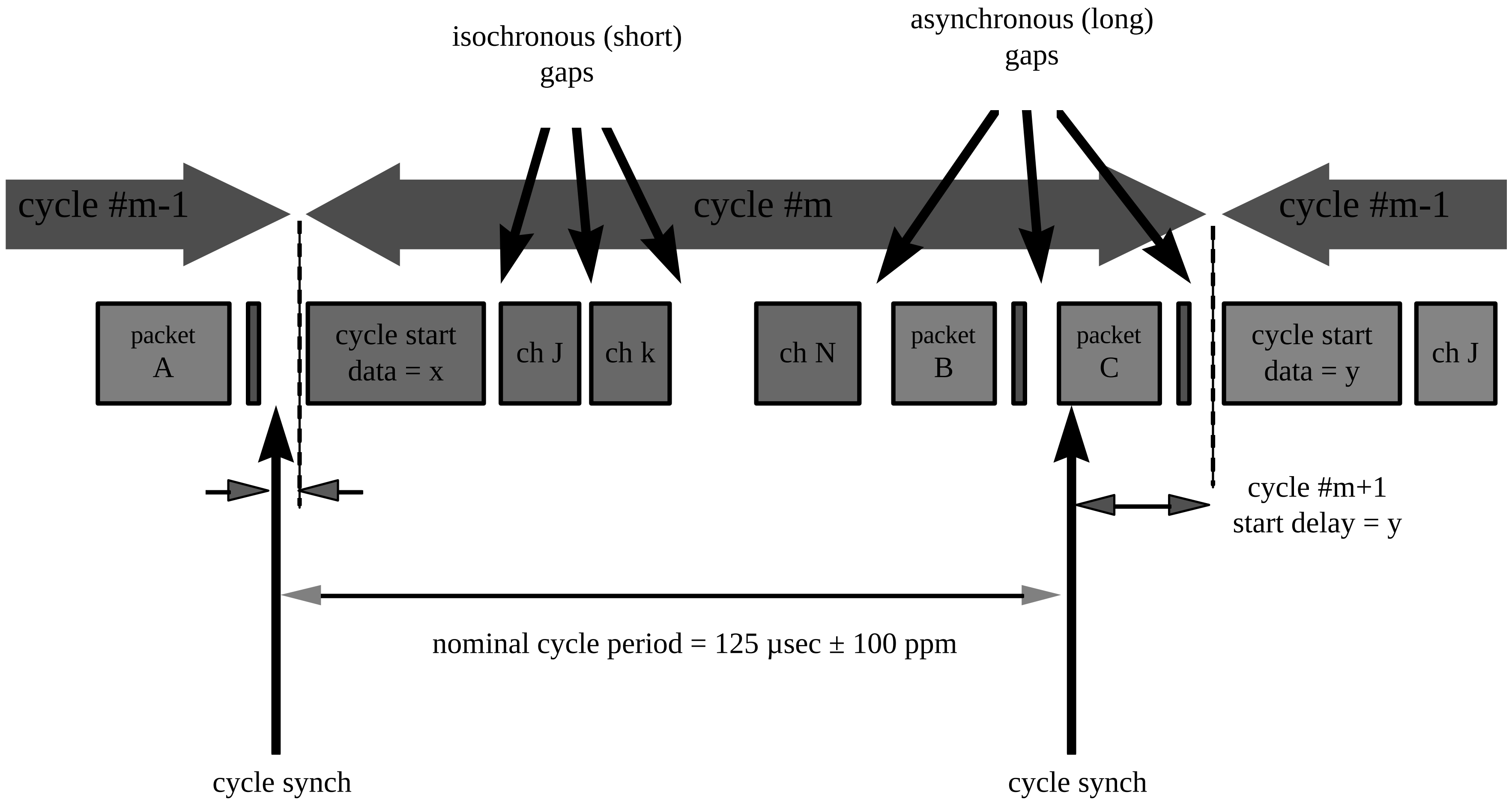
Staff Engineer

S3, Inc.

Isochronous timing

- Calculate timings for isochronous packets, and its effect on PCI FIFO requirement

Cycle Structure

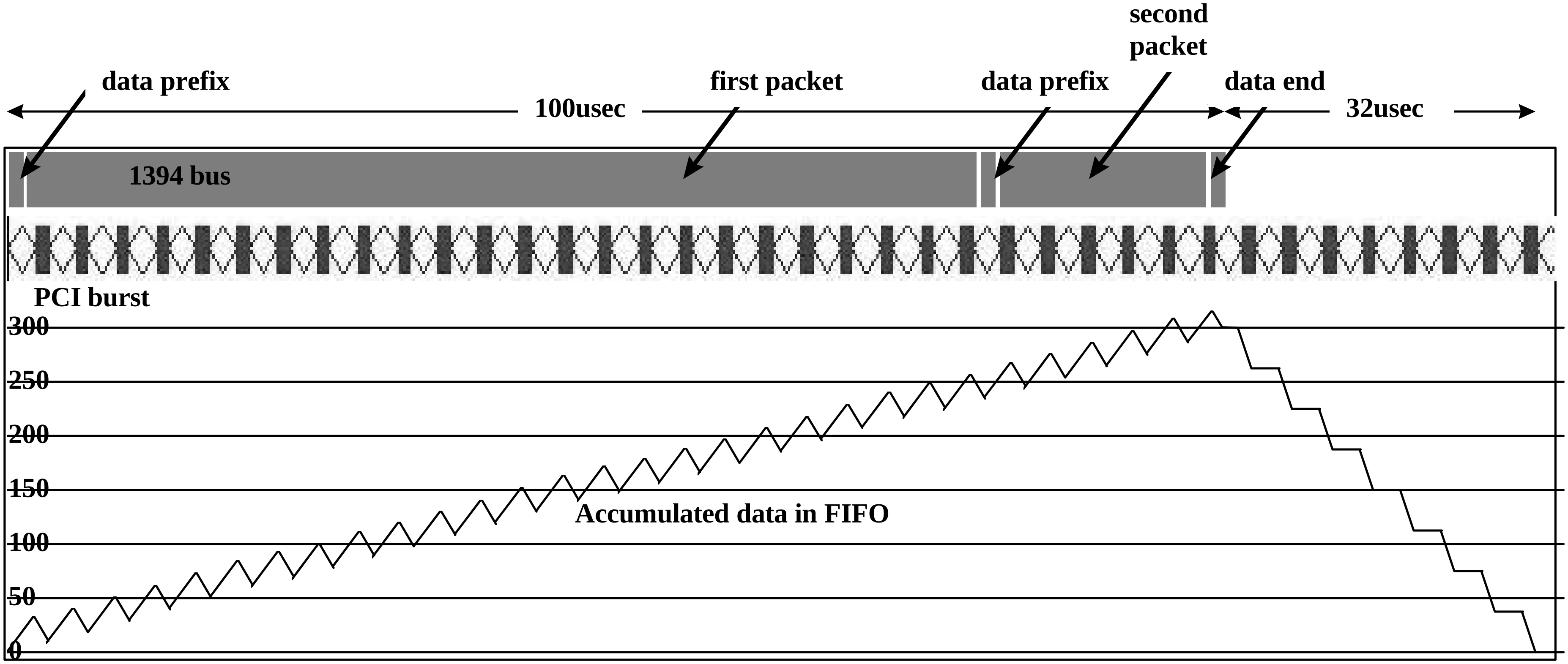


Time Components of High Efficiency Isochronous Packet

Timing parameter	time (µs)
speed_signal with data_prefix	.10
post_speed signal data_prefix	.04
Iso packet @400Mbps	83.58
max size of (4096B Data+12B Header)	
Min. packet separation	.34
Iso packet @400Mbps	
size of (758B Data+12B Header)	15.67
data end time	.24
Total	~100.00

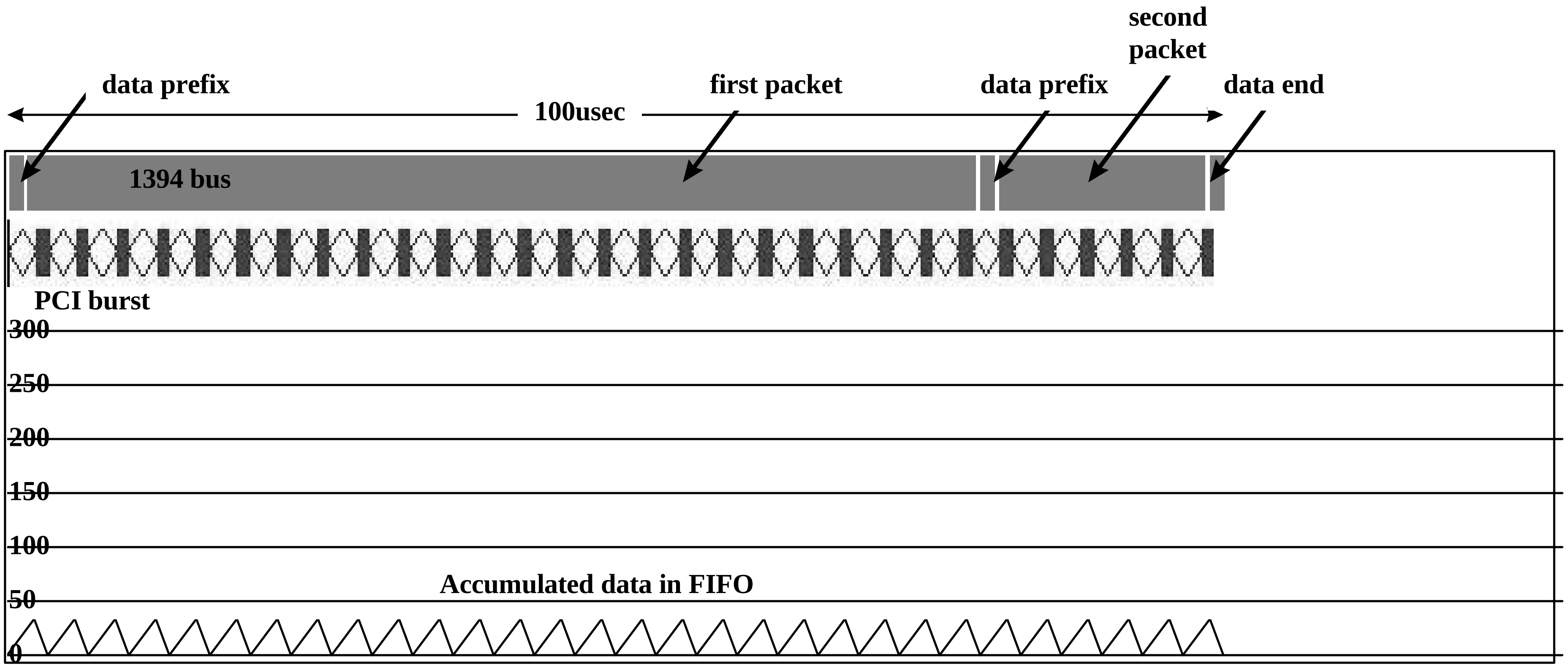
Peak bandwidth	50MB/s
Average bandwidth	48.7MB/s

PCI FIFO Size Test Case #1



- with 2.5usec arbitration latency and burst size of 32

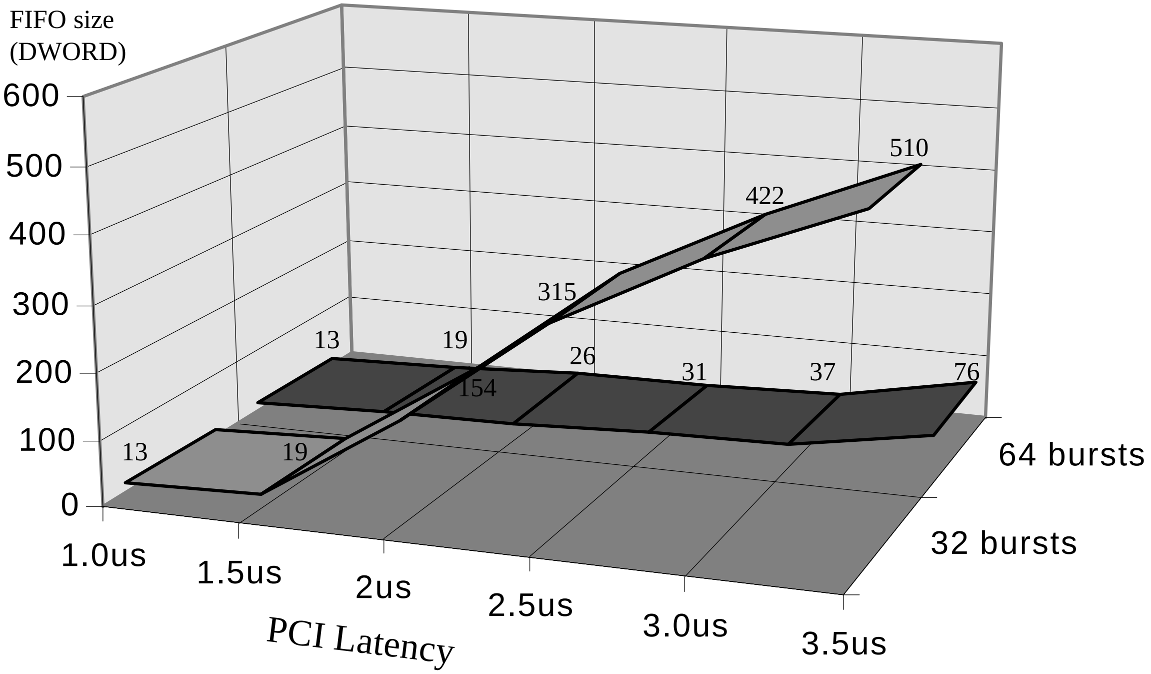
PCI FIFO Size
Test Case #2



- **with 1.6usec arbitration latency and burst size of 32**

1394 HCI OPEN

Relationship between Burst size, Latency and FIFO length



Isochronous Receive Operation

- Link receives isochronous packet and header CRC is checked OK
- Search through isochronous receive context control registers for multi-channel mode support
  - if found, check corresponding bit in the IRMultiChanMask* registers is enabled. If enabled, fetch context descriptor from CommandPtr register.

Isochronous Receive Operation

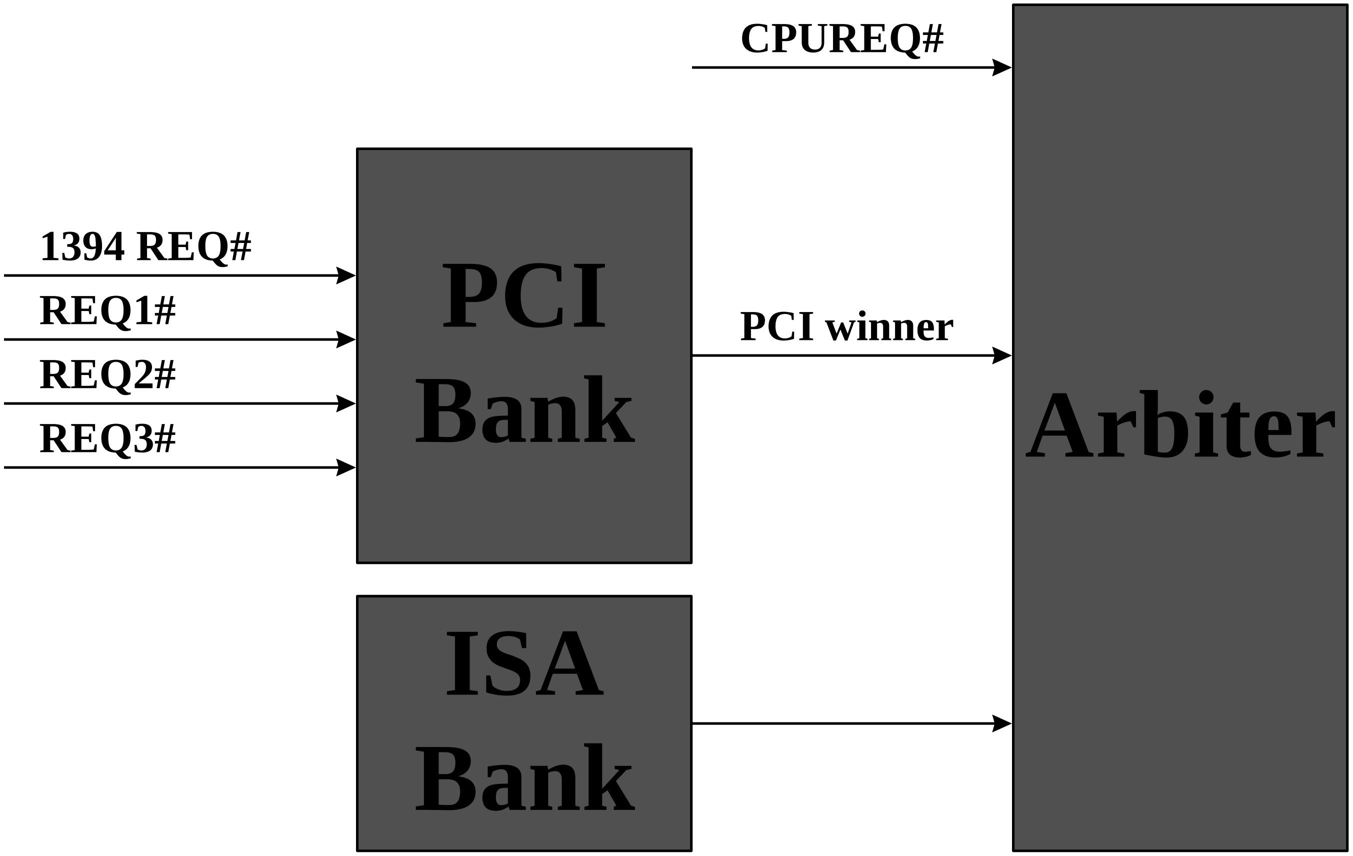
- if multiChanMode is not found, search through all IRContextMatch registers for channel number match. If match found, fetch the context descriptor from CommandPtr register. Or else, the packet is ignored.

◆ Get the dataaddress pointer from the descriptor and start unloading the data.

PCI Arbiter

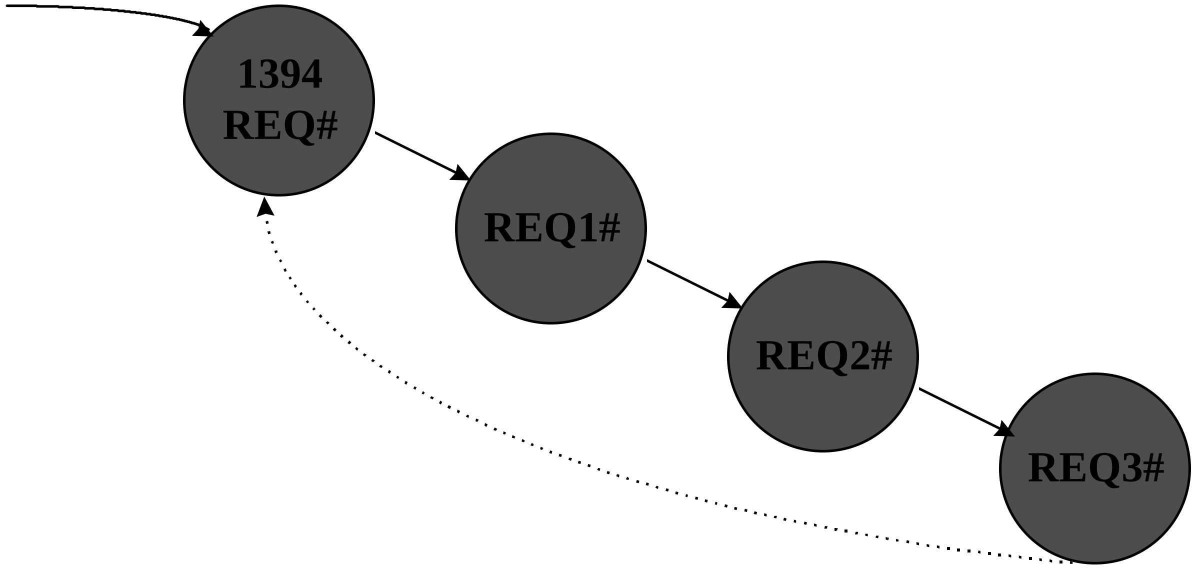
- **Implementation reflects the time to assert the GNT# signal line of a PCI bus master that asserted its REQ# signal line to request bus ownership**
- **Arbitration algorithm of the central arbiter and the priority of the PCI resources requesting bus ownership affect the FIFO requirement**
- **Predictability and low latency are important**

PCI Arbiter Implementation



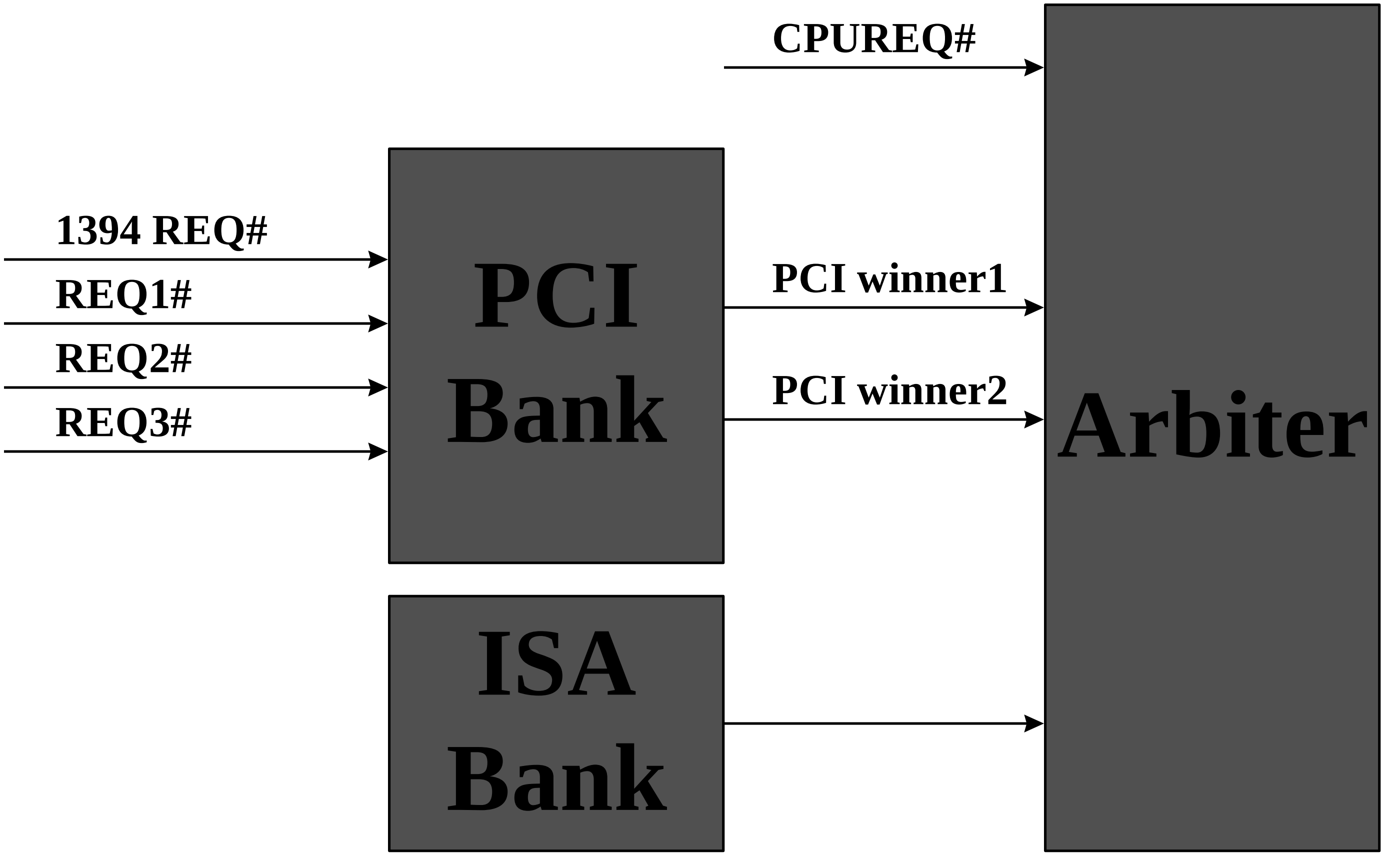
Round-robin mode

PCI Arbiter Implementation



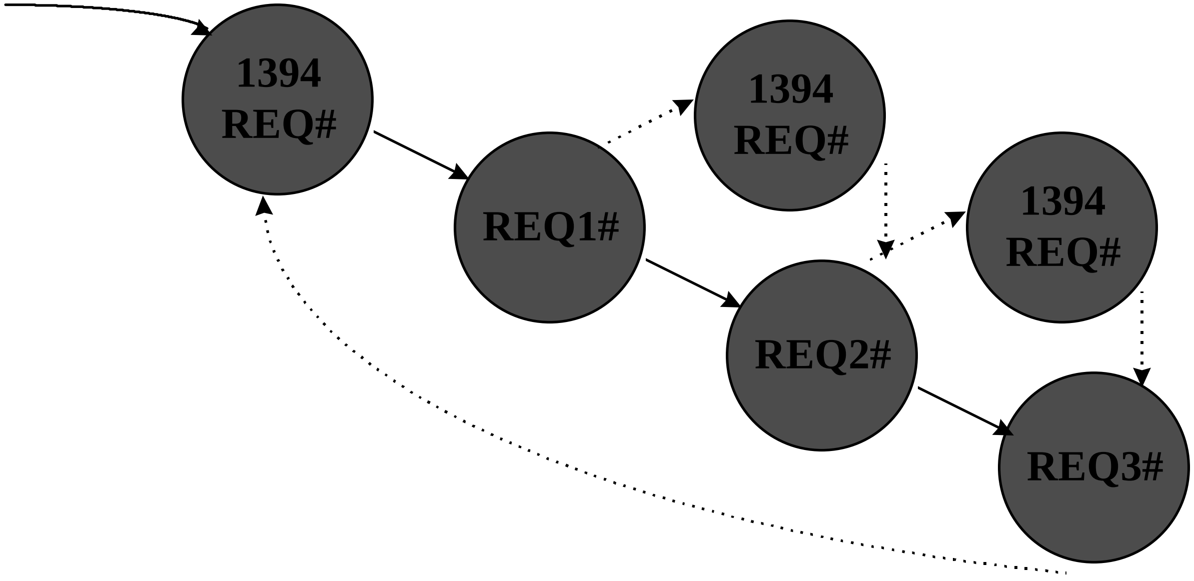
- In this environment, 1394 REQ# can win every four PCI arbitration. This gives 1394 access to PCI bus 1394 with 25% of total bandwidth

PCI Arbiter Implementation



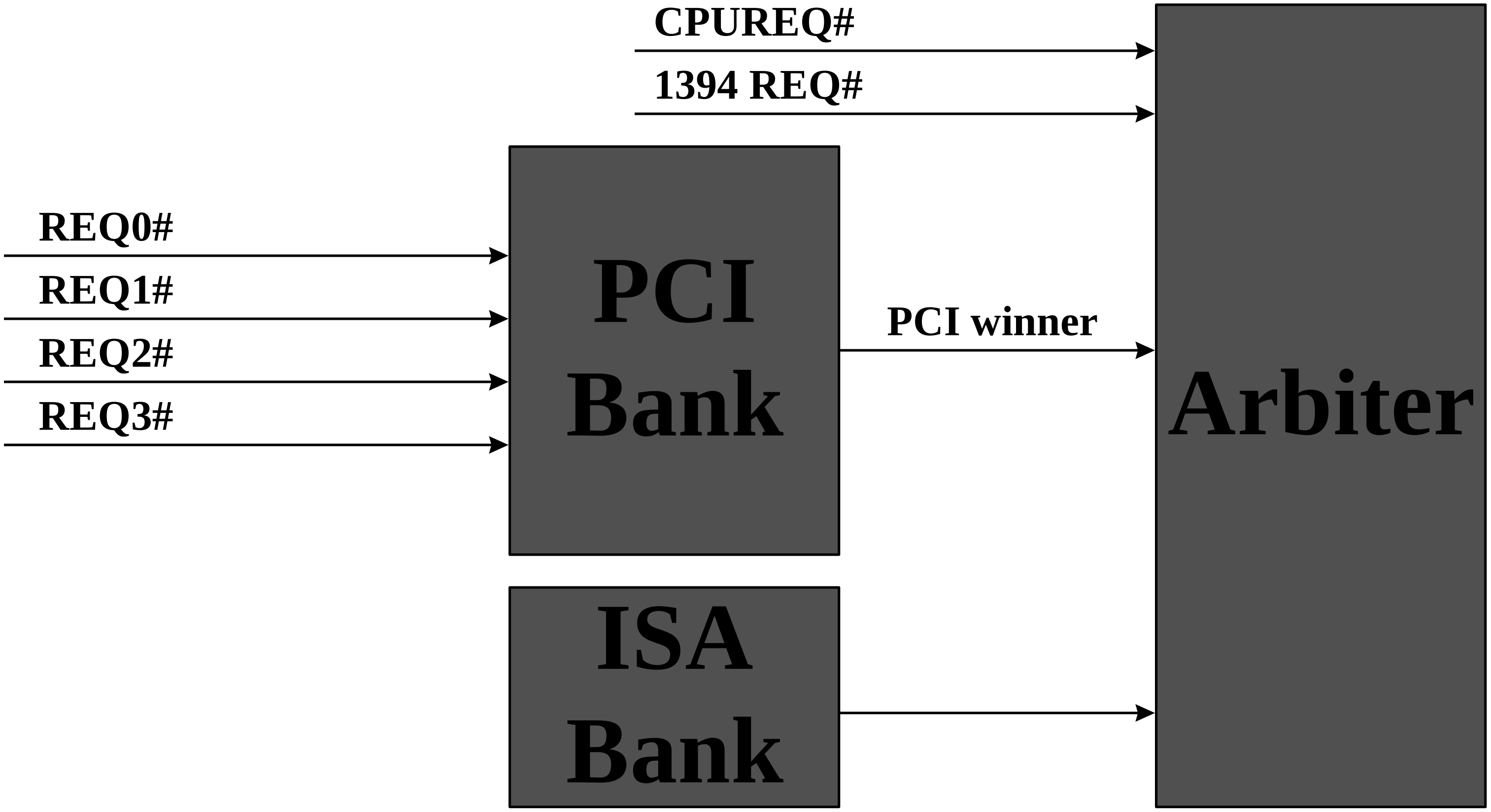
PCI high priority mode

PCI Arbiter Implementation



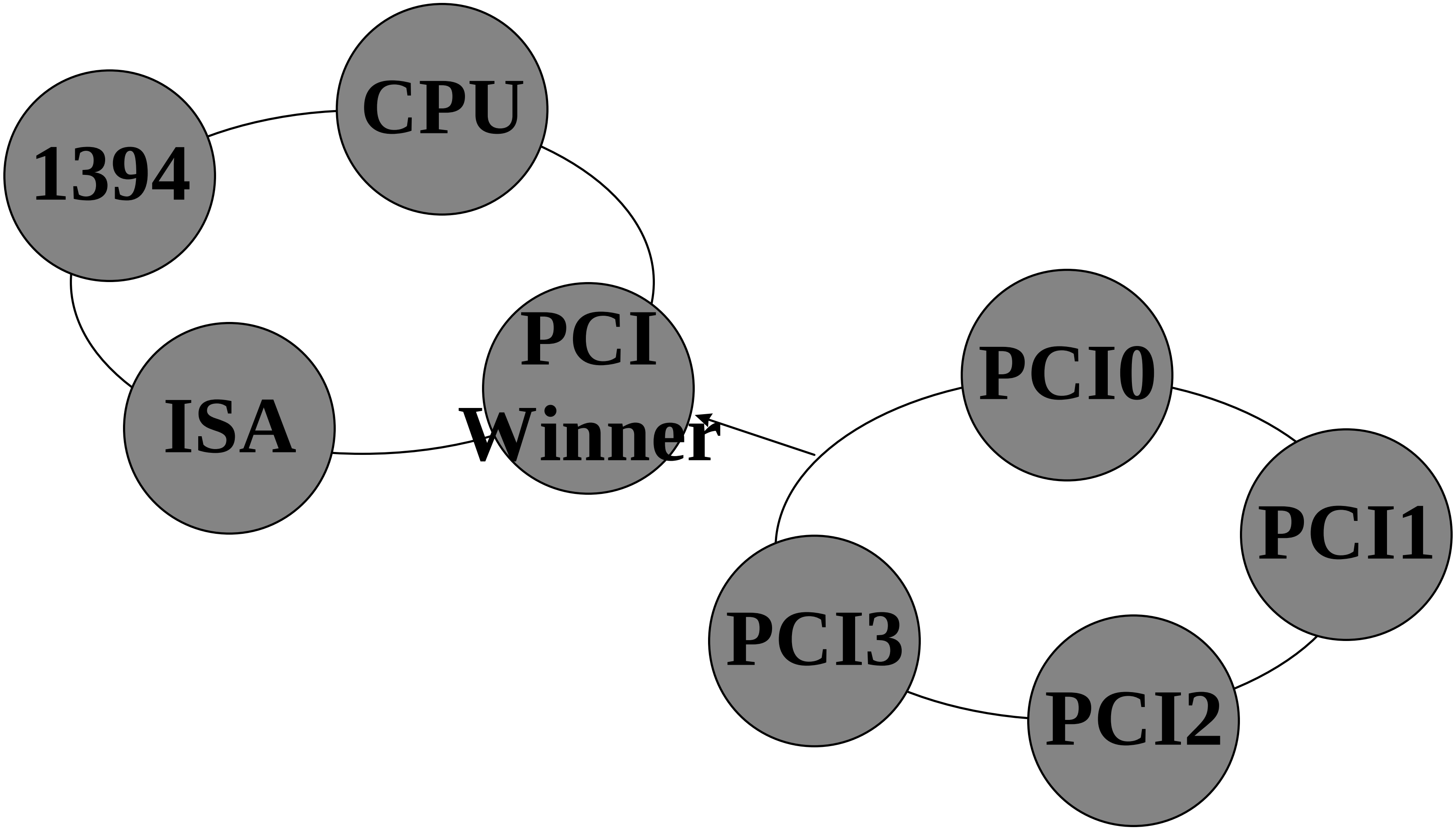
- In controlled environment, 1394 REQ# can win every other PCI arbitration. This gives priority to 1394 with 50% of total PCI bandwidth

North Bridge Arbiter Implementation

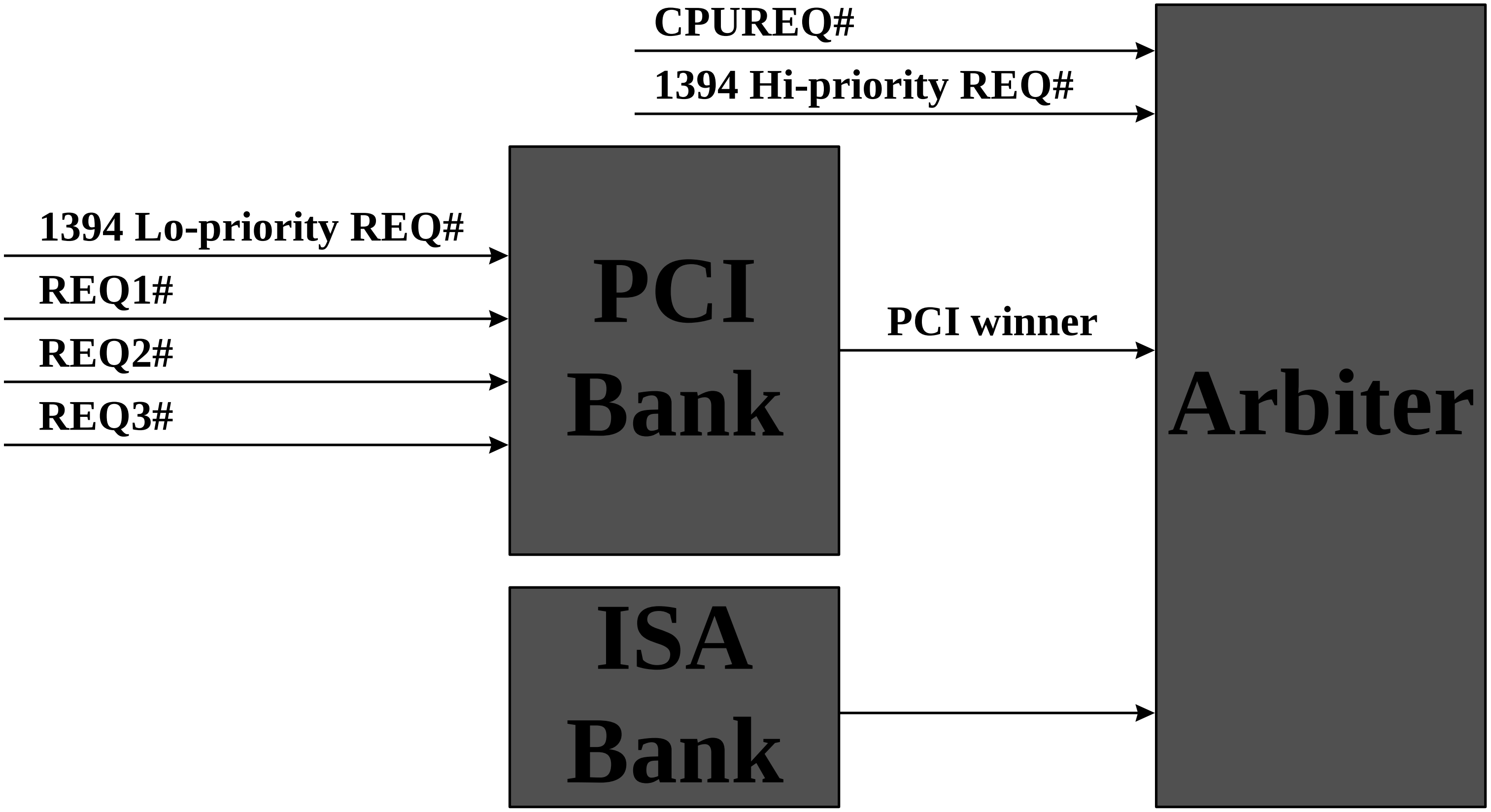


North Bridge priority mode

North Bridge Arbiter Implementation

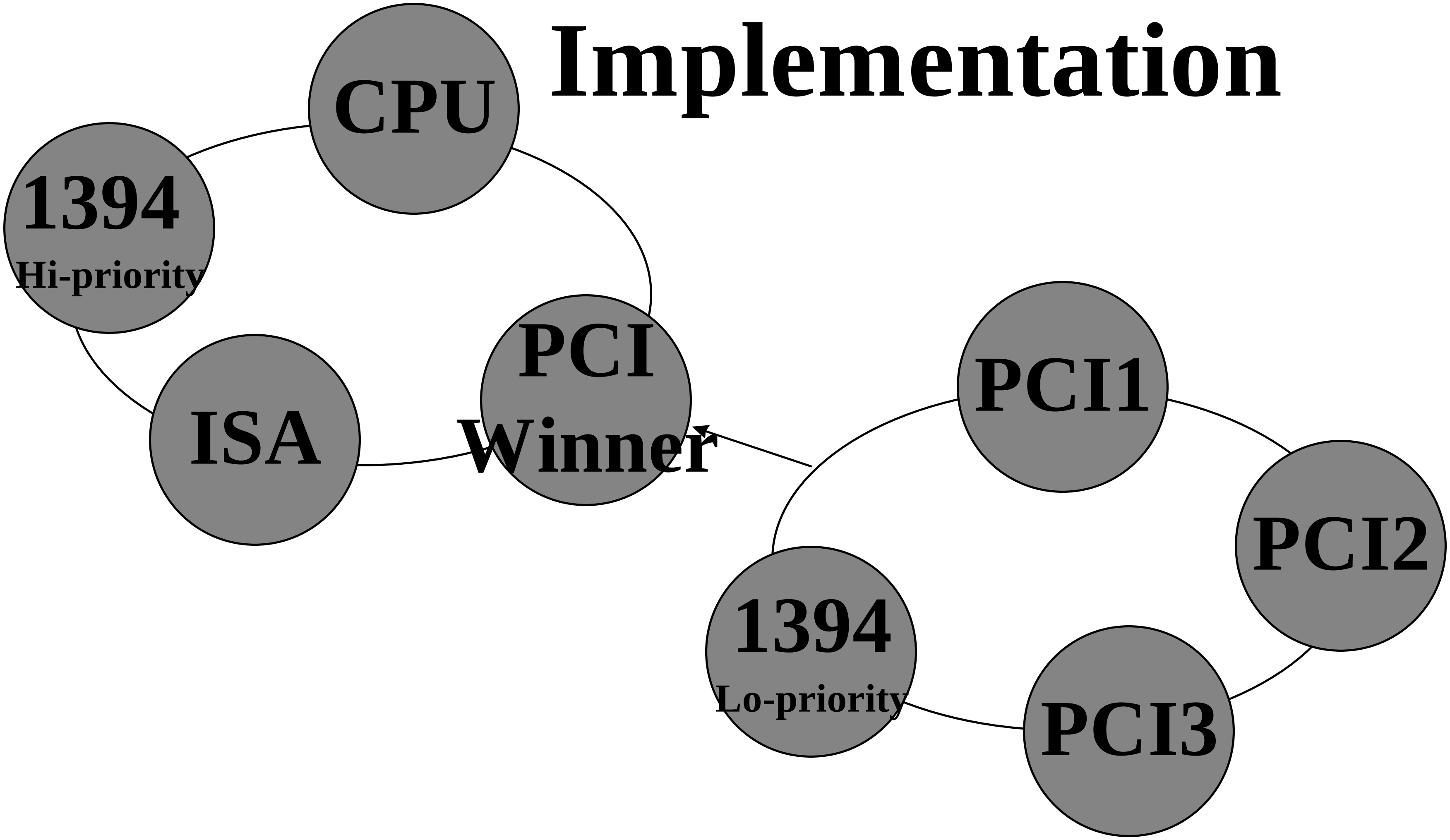


North Bridge Arbiter Implementation



North Bridge and PCI mixed priority mode

North Bridge Arbiter Implementation



High Priority Arbitrations

- Physical read and write requests if phyReqResources* are enabled to allow concatenated asynchronous subaction
- Isochronous receive for fetching input* descriptors

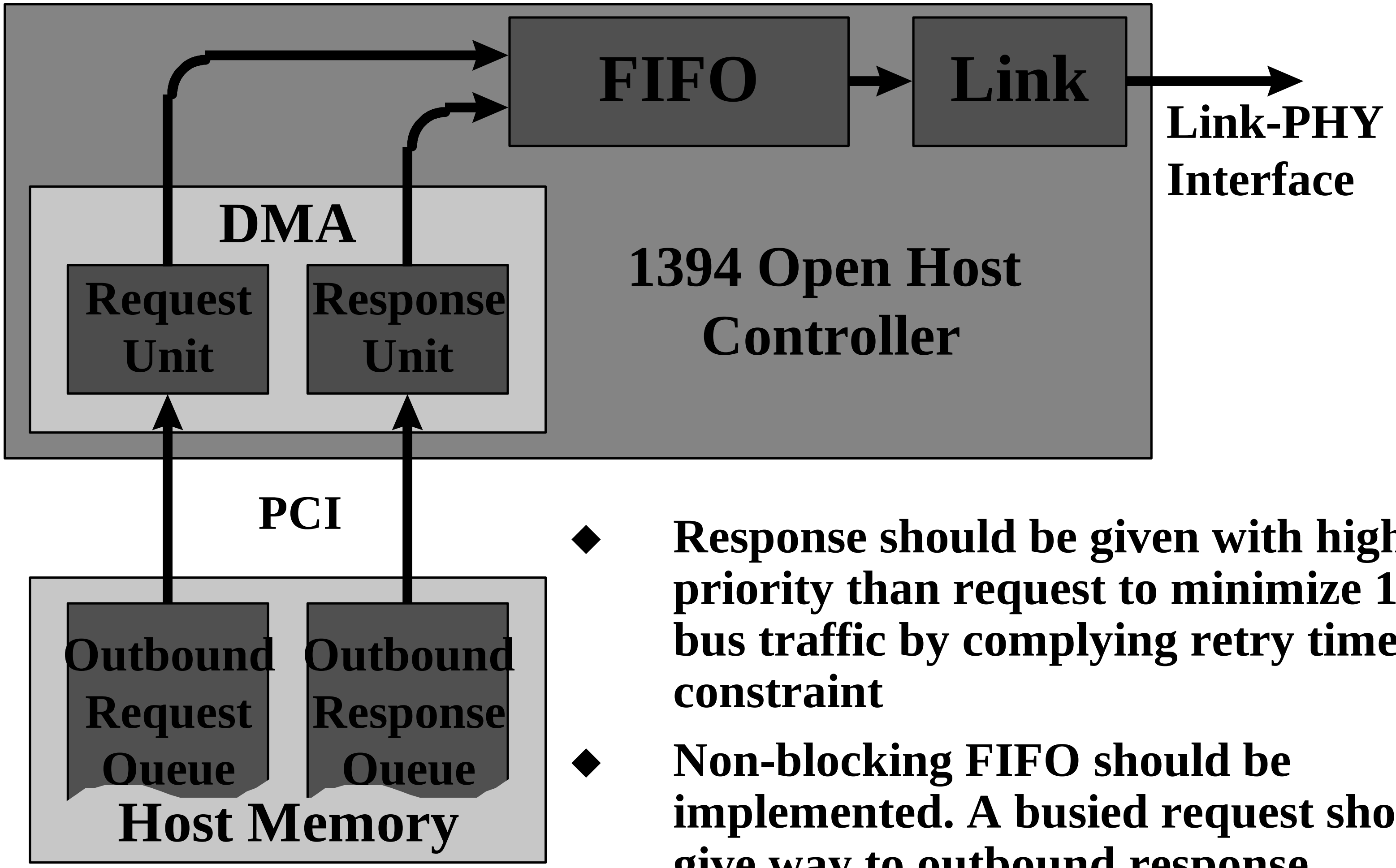
PCI Features Needed

- FIFO requirement depends on
  - long bursts
  - fastest possible TRDY# as a target
  - continuous IRDY# as initiator
  - fast back-to-back
  - eliminates idle cycle following a write
  - memory write and invalidate eliminates system cache snoop
  - memory read line multiple
  - optimize system memory pre-fetching

PCI Features Needed

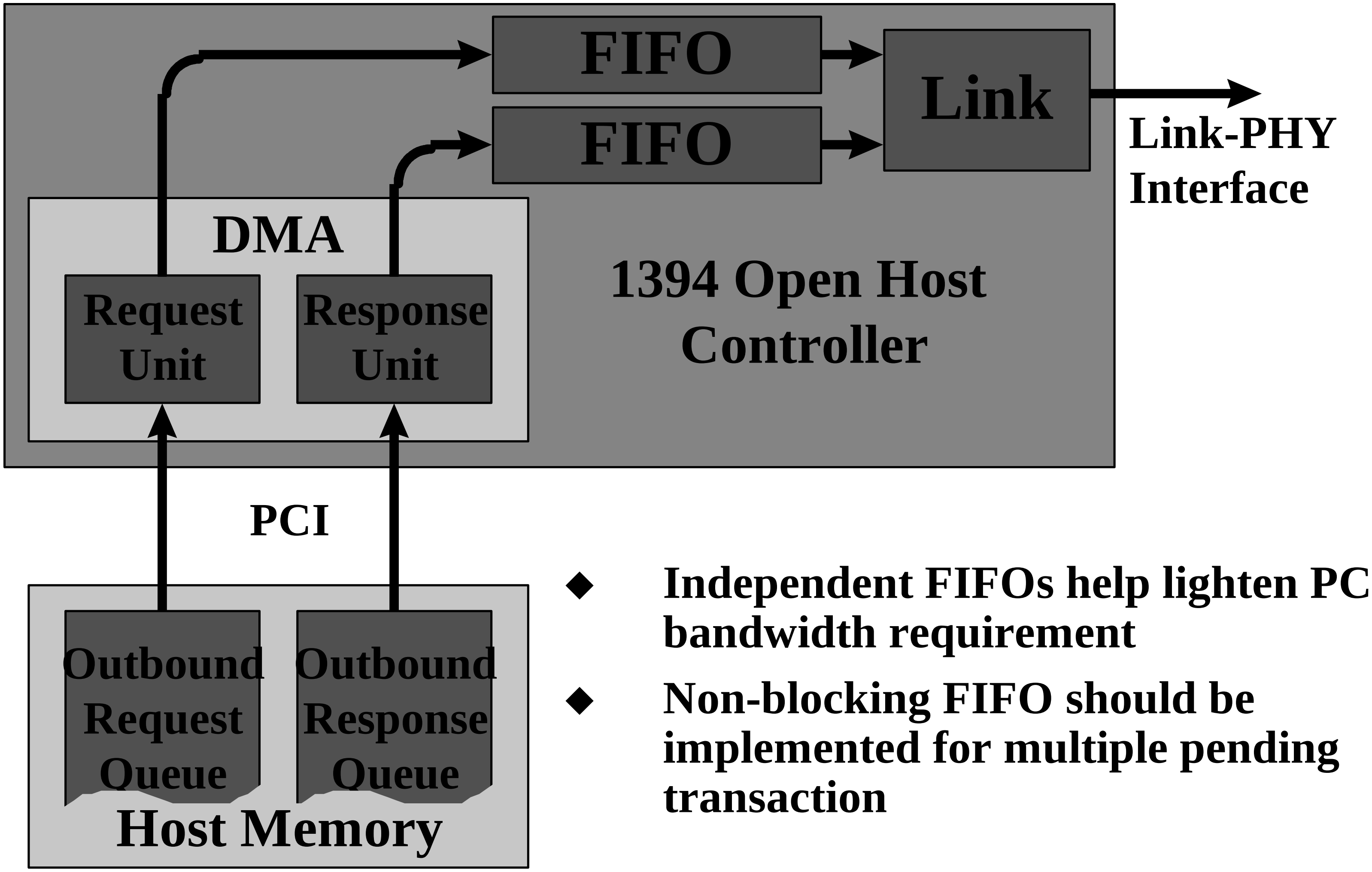
- **FIFO requirement depends on**
  - **Corelogic should allow bus master to perform more than one transaction per arbitration**
    - **This allows the arbiter to maintain ownership of the bus after completion of the current transaction. This is particular useful for continuous operation during descriptor fetching and data moving.**

PCI Implementations



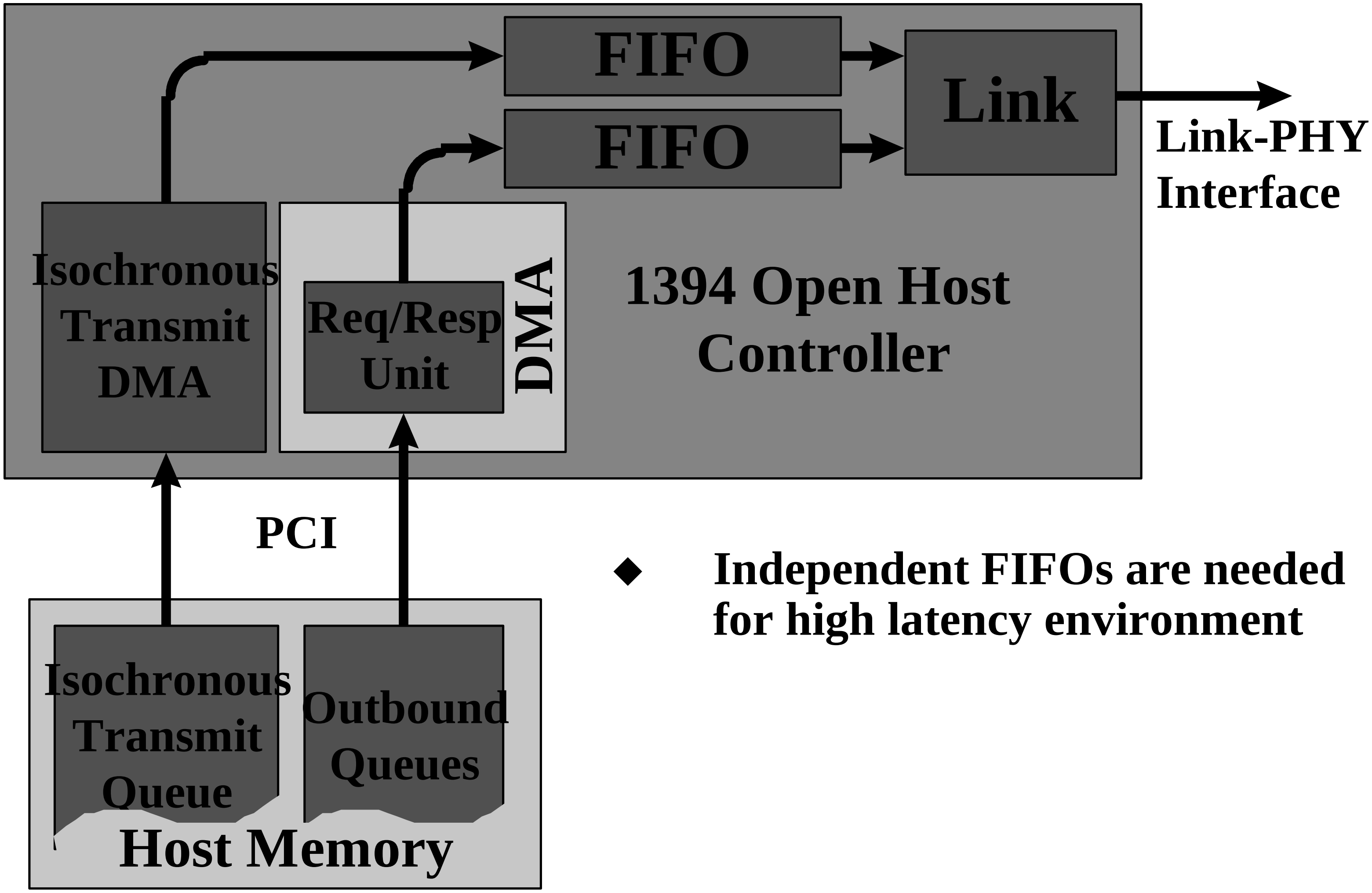
- Response should be given with higher priority than request to minimize 1394 bus traffic by complying retry timeout constraint
- Non-blocking FIFO should be implemented. A busied request should give way to outbound response

PCI Implementations



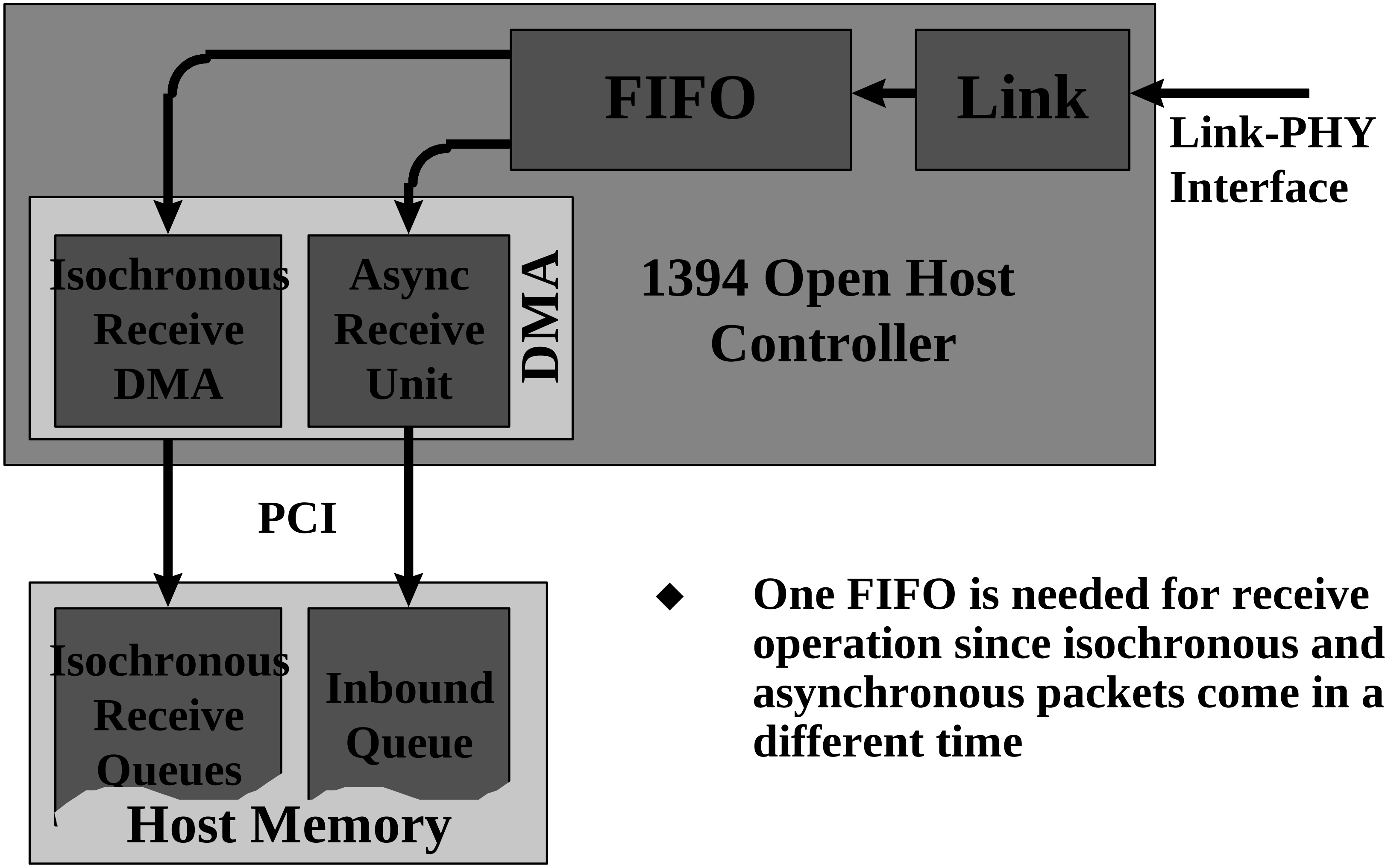
- Independent FIFOs help lighten PCI bandwidth requirement
- Non-blocking FIFO should be implemented for multiple pending transaction

PCI Implementations



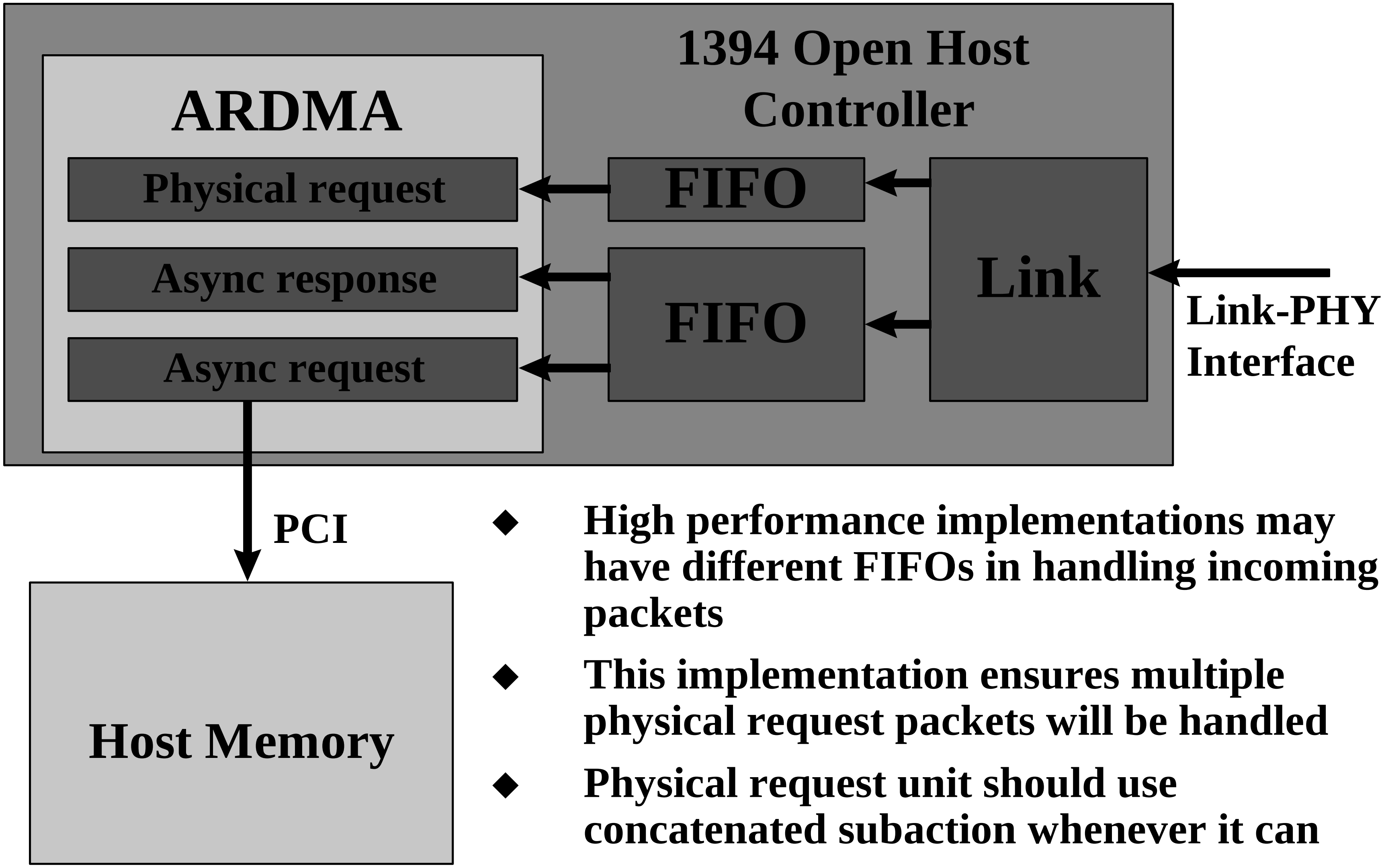
- Independent FIFOs are needed for high latency environment

PCI Implementations



- One FIFO is needed for receive operation since isochronous and asynchronous packets come in at different time

PCI Implementations



- High performance implementations may have different FIFOs in handling incoming packets
- This implementation ensures multiple physical request packets will be handled
- Physical request unit should use concatenated subaction whenever it can

Bus Management CSR Register Operation

- To update the Bus Management CSR registers:
  - updates csrData with the desired value
  - updates csrCompare with the current value
  - updates csrSel with proper offset
  - the csrData should have the old content as the one updated in the csrCompare value if compare-swap operation is successful when csrDone is set.

Bus Management CSR Register Operation

- To read the Bus Management CSR registers
  - updates the csrData and csrCompare with the same arbitrary values
  - updates the csrSel with proper offset
  - the value is returned on the csrData if csrDone is set

Open HCI Compliant P1394A Requirements

- **Disable control per port**

  This ensures the Open HCI systems is safe from the external environment by running only on the internal nodes

Open HCI Compliant P1394A Requirements

- During self-ID process, the maximum physical_ID remains at 63 if more than 63 nodes are on the local bus. No roll over of physical_ID is allowed.
- Any PHY of physical_ID of 63 will ignore link-on and PHY-configuration packet.
- Link of physical_ID of 63 will not send any packet

Open HCI Compliant P1394A Requirements

- Bus manager algorithm must support 3 bit speed code.
- If LK_EVENT.ind(cycle_too_long) is set, CONTROL.cyclemaster is cleared.
- The cycle_too_long is defined as since the last cycle start packet was received, an 125usec has passed, and a subaction gap was not yet detected.

Open HCI Compliant P1394A Requirements

- A timeout interrupt for PHY register read shall be implemented.
- Response in first try need not to follow the fairness protocol

BUS INFO BLOCK Implementations

- BUS INFO BLOCK is loaded by:
  - Hardware - loaded after chip reset by serial or parallel ROM.
  - Software - loaded during execution of boot code

BUS INFO BLOCK Implementations

- Global Unique ID may only be written once either by hardware or software and shall be write-protected afterward
- No linkEnable shall be enabled before the Global Unique ID is written

Source node’s BUS ID in Response Packet

- Two BUS IDs may be used from requesters to identify the node
- 10’h3FF (local-bus) or Specific BUS ID assigned from bus manager
- Same ID shall be used for read response packet as was used in the request packet

Source node's BUS ID in Response Packet

- Bit 23 SrcBusID of the first quadlet in the packet header is provided to the link layer on which BUS ID to be used in the response packets
  - 0 - 10'h3FF is used
  - 1 - NodeID.BusNumber is used

Other Implementation Details

- OpenHCI implements only CSR registers of BUS MANAGEMENT CSR, and BUS INFO BLOCK, NODE_IDS in hardware. All other request packets shall be sent to asynchronous receive or physical request DMA accordingly
- Only quadlet read/compare swap within HCI implemented CSR registers are allowed. Otherwise, type error in response code is returned.

End