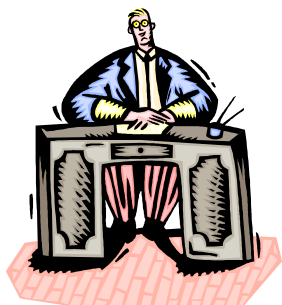
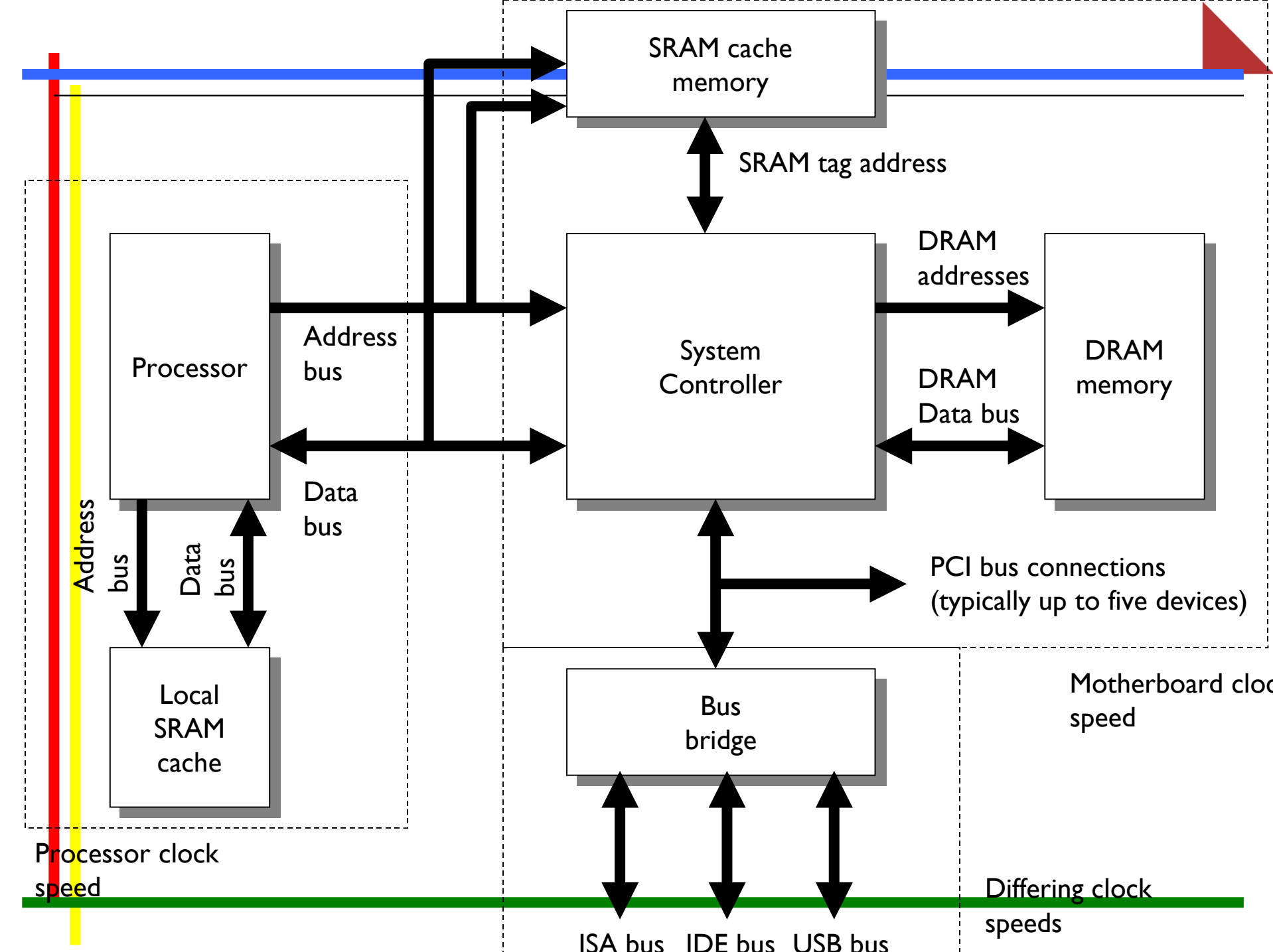
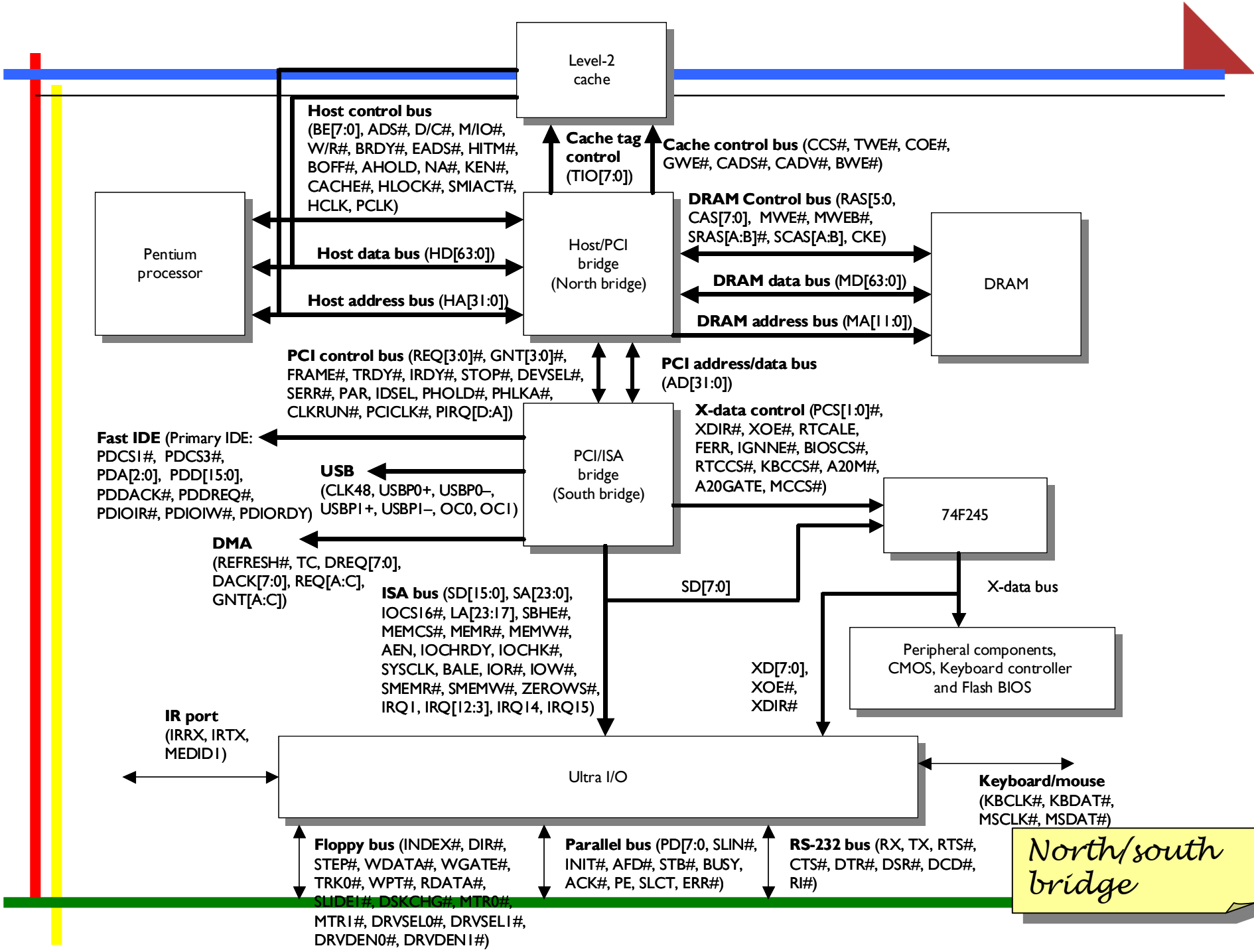
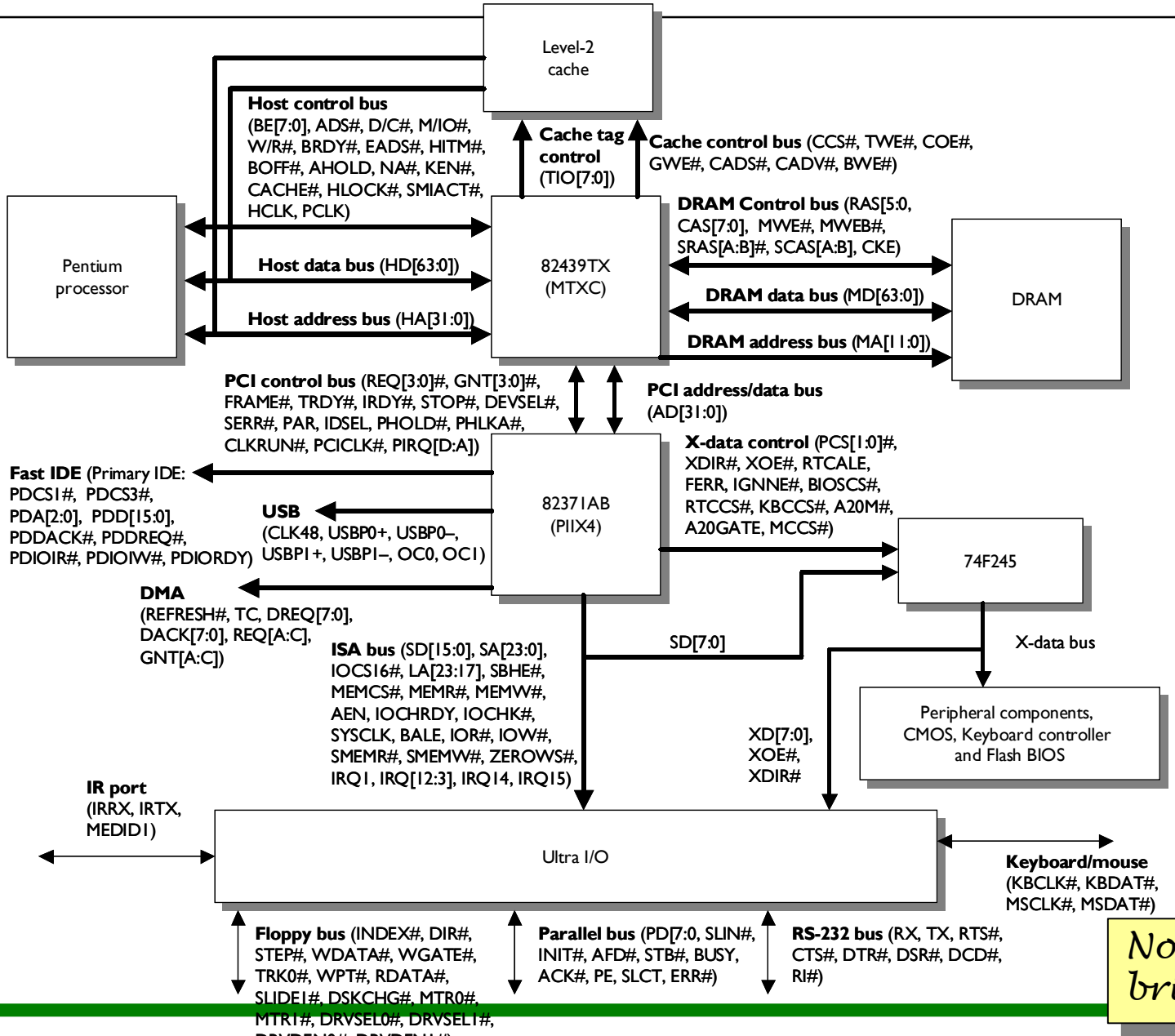
PCI Bus

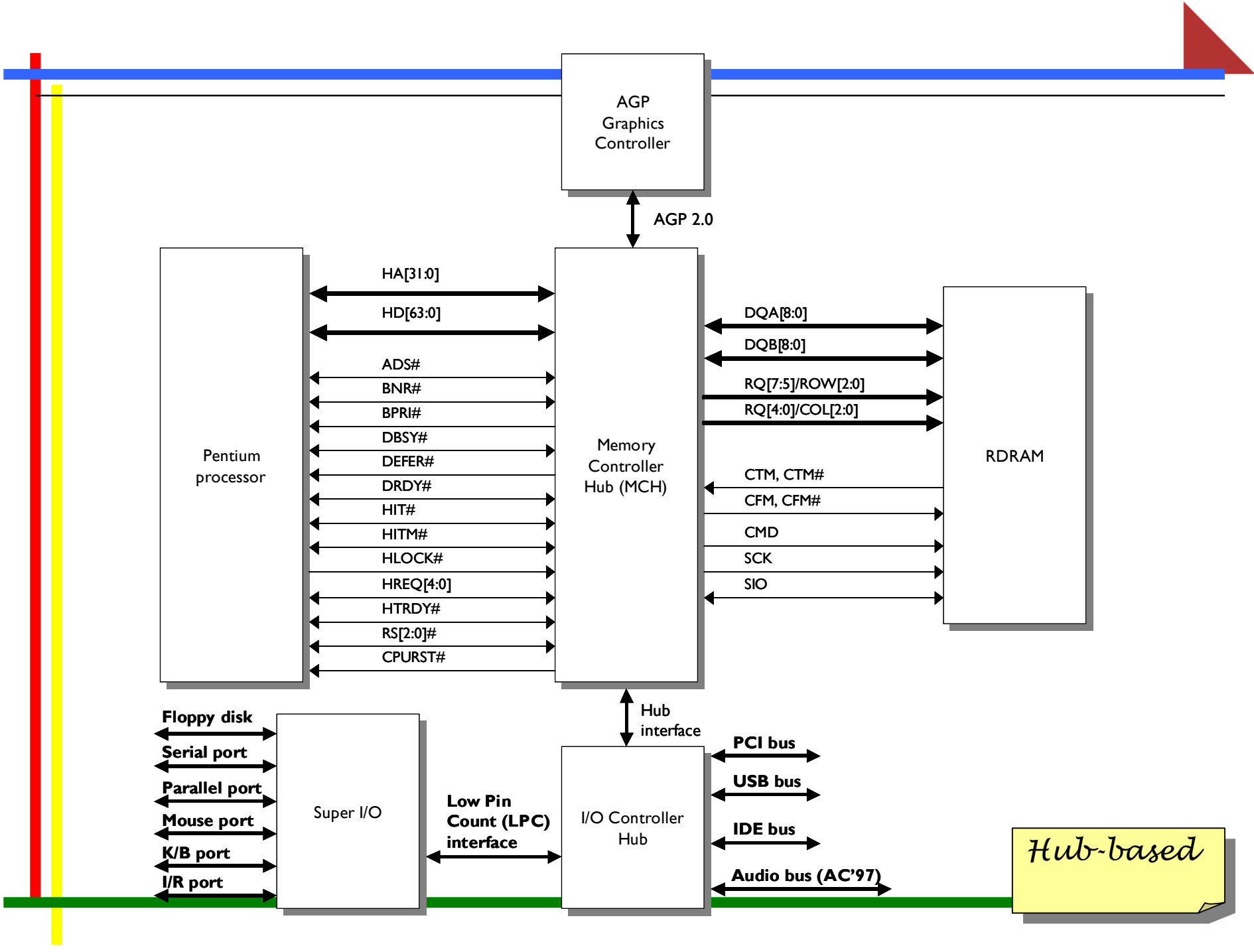
W.J. BUCHANAN, *Napier University, Edinburgh, UK.*

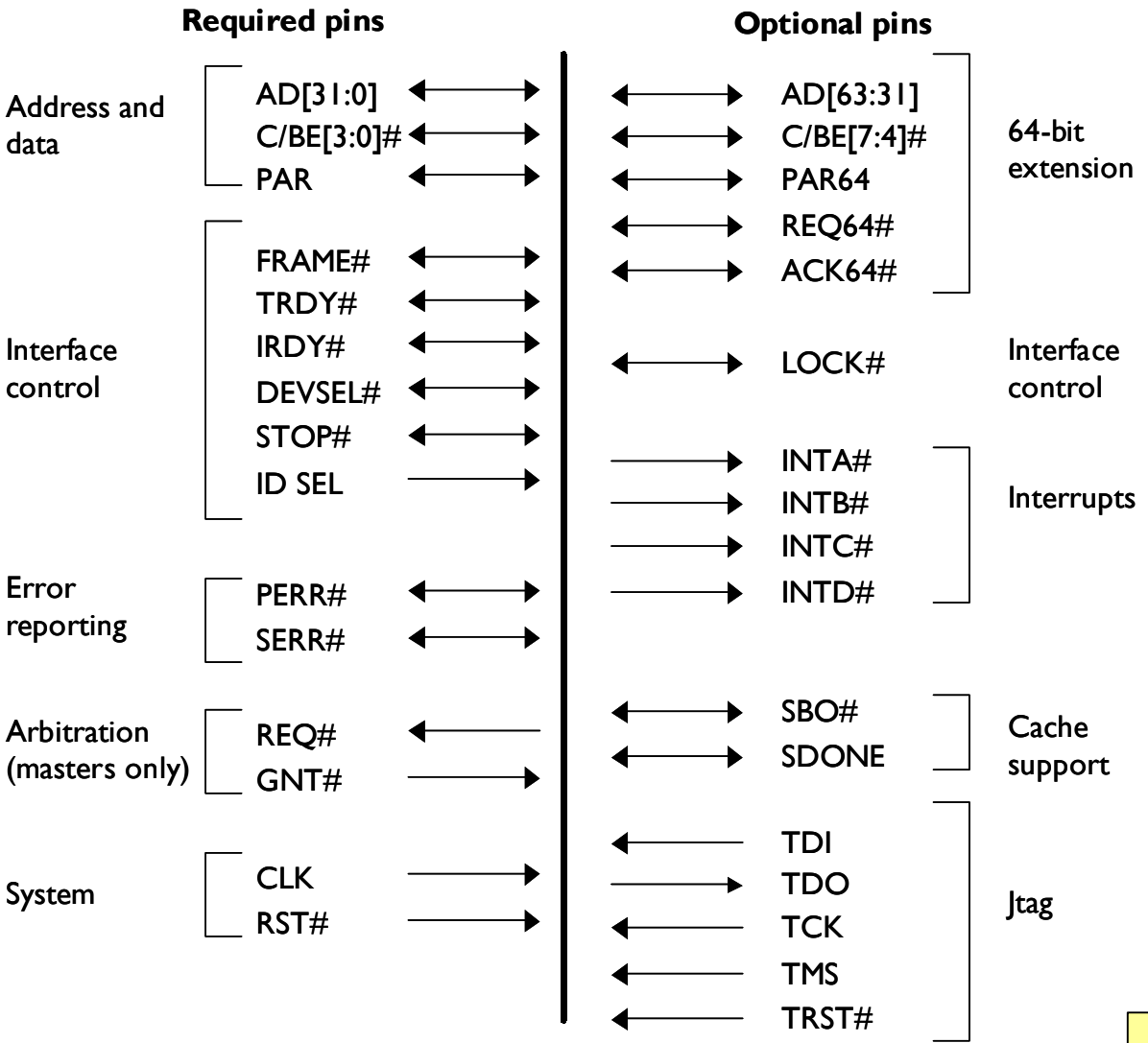



SRAM cache memory
SRAM tag address
Processor
Address bus
Data bus
System Controller
DRAM addresses
DRAM Data bus
DRAM memory
Address bus
Data bus
Local SRAM cache
PCI bus connections (typically up to five devices)
Bus bridge
Motherboard clock speed
Processor clock speed
Differing clock speeds
ISA bus
IDE bus
USB bus

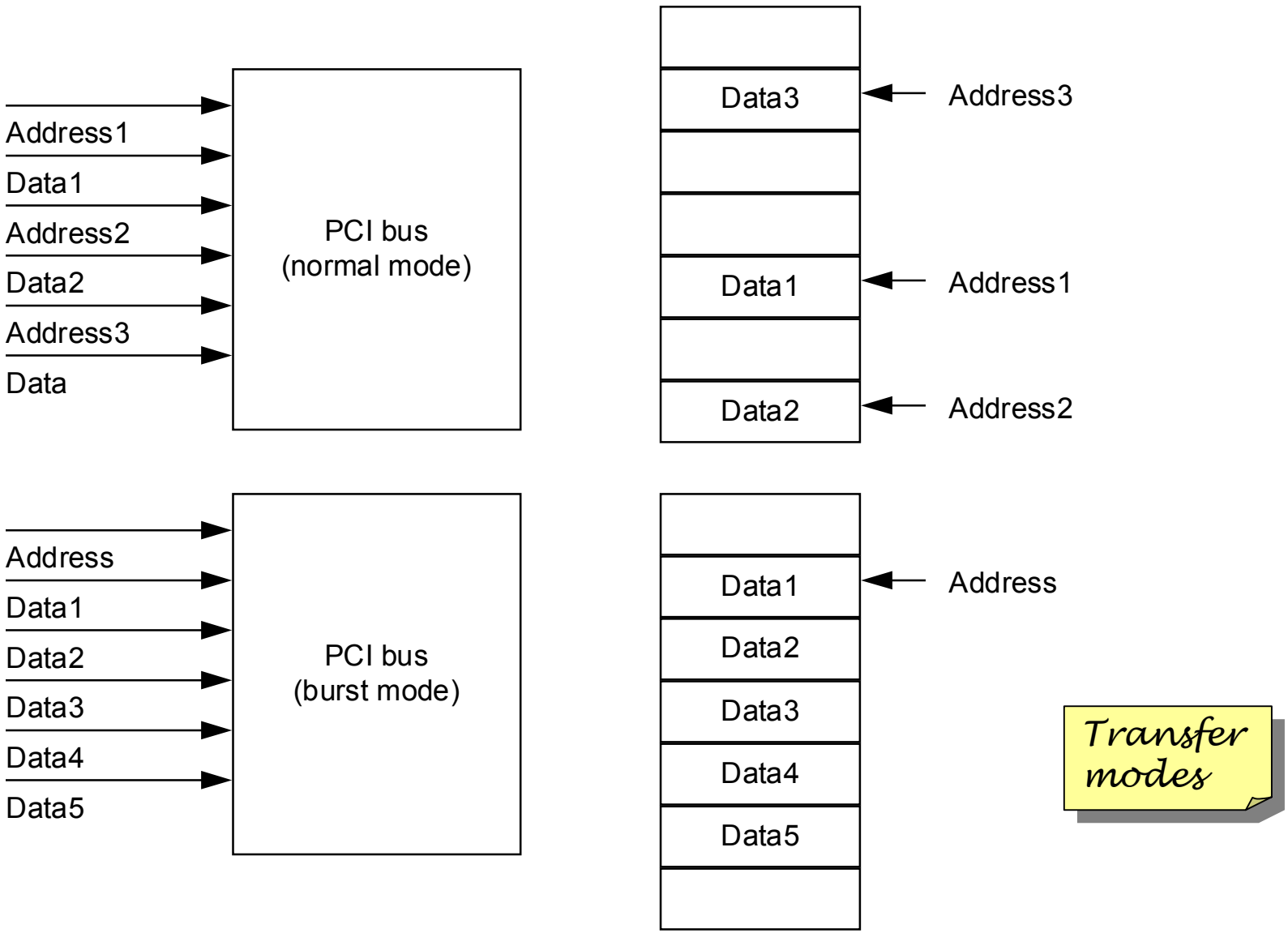

Level-2 cache
Host control bus (BE[7:0], ADS#, D/C#, M/IO#, W/R#, BRDY#, EADS#, HITM#, BOFF#, AHOLD, NA#, KEN#, CACHE#, HLOCK#, SMIACT#, HCLK, PCLK)
Cache tag control (TIO[7:0])
Cache control bus (CCS#, TWE#, COE#, GWE#, CADS#, CADV#, BWE#)
Pentium processor
Host data bus (HD[63:0])
Host address bus (HA[31:0])
Host/PCI bridge (North bridge)
DRAM Control bus (RAS[5:0, CAS[7:0], MWE#, MWEB#, SRAS[A:B]#, SCAS[A:B], CKE)
DRAM data bus (MD[63:0])
DRAM address bus (MA[11:0])
DRAM
PCI control bus (REQ[3:0]#, GNT[3:0]#, FRAME#, TRDY#, IRDY#, STOP#, DEVSEL#, SERR#, PAR, IDSEL, PHOLD#, PHLKA#, CLKRUN#, PCICLK#, PIRQ[D:A])
PCI address/data bus (AD[31:0])
Fast IDE (Primary IDE: PDCS1#, PDCS3#, PDA[2:0], PDD[15:0], PDDACK#, PDDREQ#, PDIOIR#, PDIOIW#, PDIORDY)
USB (CLK48, USBP0+, USBP0–, USBP1+, USBP1–, OC0, OC1)
DMA (REFRESH#, TC, DREQ[7:0], DACK[7:0], REQ[A:C], GNT[A:C])
PCI/ISA bridge (South bridge)
X-data control (PCS[1:0]#, XDIR#, XOE#, RTCALE, FERR, IGNNE#, BIOSCS#, RTCCS#, KBCCS#, A20M#, A20GATE, MCCS#)
74F245
SD[7:0]
X-data bus
ISA bus (SD[15:0], SA[23:0], IOCS16#, LA[23:17], SBHE#, MEMCS#, MEMR#, MEMW#, AEN, IOCHRDY, IOCHK#, SYSCLK, BALE, IOR#, IOW#, SMEMR#, SMEMW#, ZEROWS#, IRQ1, IRQ[12:3], IRQ14, IRQ15)
Peripheral components, CMOS, Keyboard controller and Flash BIOS
XD[7:0], XOE#, XDIR#
IR port (IRRX, IRTX, MEDID1)
Ultra I/O
Keyboard/mouse (KBCLK#, KBDAT#, MSCLK#, MSDAT#)
Floppy bus (INDEX#, DIR#, STEP#, WDATA#, WGATE#, TRK0#, WPT#, RDATA#, SLIDE1#, DSKCHG#, MTR0#, MTR1#, DRVSEL0#, DRVSEL1#, DRVDEN0#, DRVDEN1#)
Parallel bus (PD[7:0, SLIN#, INIT#, AFD#, STB#, BUSY, ACK#, PE, SLCT, ERR#)
RS-232 bus (RX, TX, RTS#, CTS#, DTR#, DSR#, DCD#, RI#)
North/south bridge

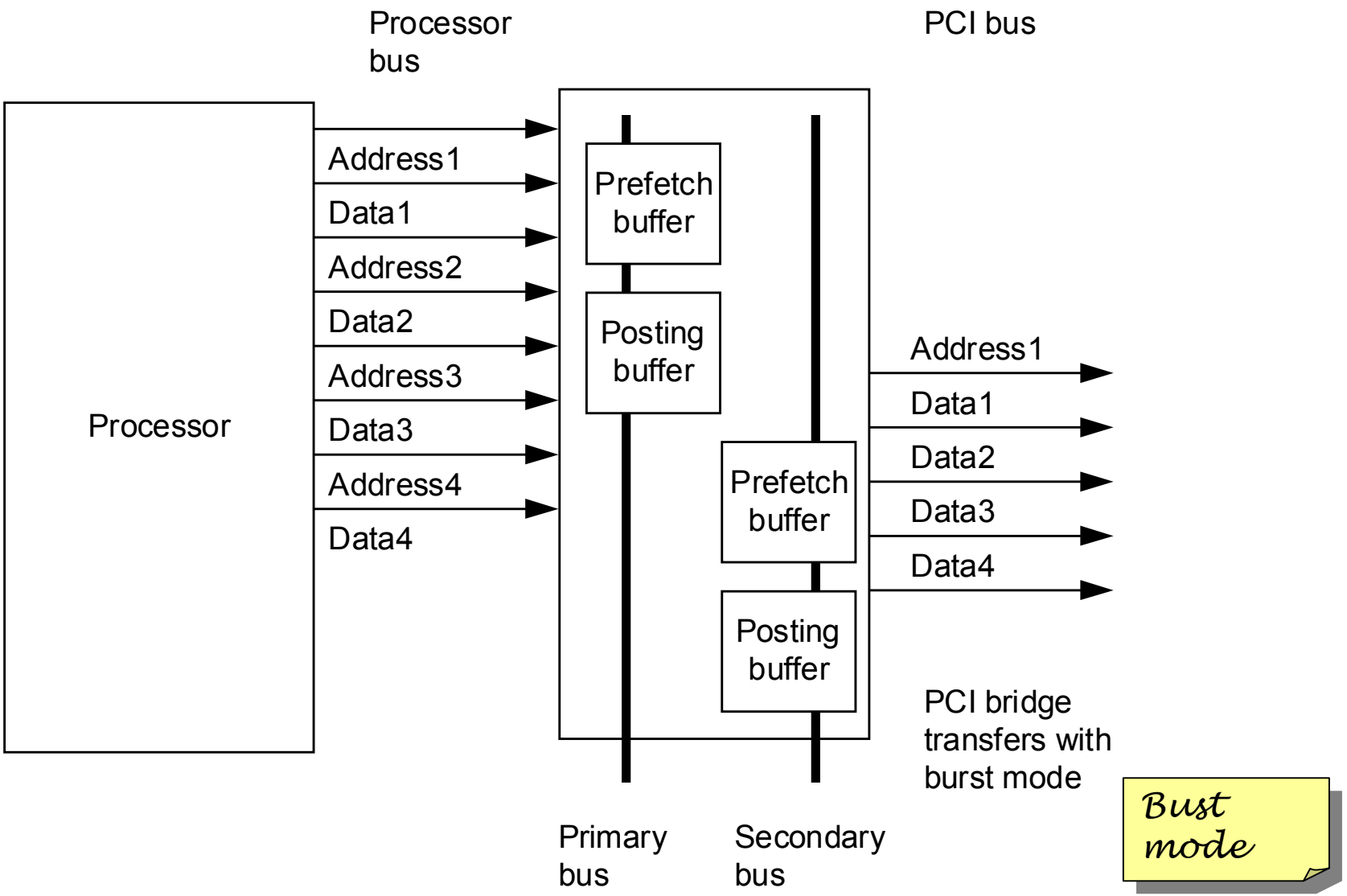

Level-2 cache
Host control bus (BE[7:0], ADS#, D/C#, M/IO#, W/R#, BRDY#, EADS#, HITM#, BOFF#, AHOLD, NA#, KEN#, CACHE#, HLOCK#, SMIACT#, HCLK, PCLK)
Cache tag control (TIO[7:0])
Cache control bus (CCS#, TWE#, COE#, GWE#, CADS#, CADV#, BWE#)
Pentium processor
Host data bus (HD[63:0])
Host address bus (HA[31:0])
82439TX (MTXC)
DRAM Control bus (RAS[5:0, CAS[7:0], MWE#, MWEB#, SRAS[A:B]#, SCAS[A:B], CKE)
DRAM data bus (MD[63:0])
DRAM address bus (MA[11:0])
DRAM
PCI control bus (REQ[3:0]#, GNT[3:0]#, FRAME#, TRDY#, IRDY#, STOP#, DEVSEL#, SERR#, PAR, IDSEL, PHOLD#, PHLKA#, CLKRUN#, PCICLK#, PIRQ[D:A])
PCI address/data bus (AD[31:0])
Fast IDE (Primary IDE: PDCS1#, PDCS3#, PDA[2:0], PDD[15:0], PDDACK#, PDDREQ#, PDIOIR#, PDIOIW#, PDIORDY)
USB (CLK48, USBP0+, USBP0–, USBP1+, USBP1–, OC0, OC1)
82371AB (PIIX4)
X-data control (PCS[1:0]#, XDIR#, XOE#, RTCALE, FERR, IGNNE#, BIOSCS#, RTCCS#, KBCCS#, A20M#, A20GATE, MCCS#)
74F245
DMA (REFRESH#, TC, DREQ[7:0], DACK[7:0], REQ[A:C], GNT[A:C])
SD[7:0]
X-data bus
ISA bus (SD[15:0], SA[23:0], IOCS16#, LA[23:17], SBHE#, MEMCS#, MEMR#, MEMW#, AEN, IOCHRDY, IOCHK#, SYSCLK, BALE, IOR#, IOW#, SMEMR#, SMEMW#, ZEROWS#, IRQ1, IRQ[12:3], IRQ14, IRQ15)
Peripheral components, CMOS, Keyboard controller and Flash BIOS
XD[7:0], XOE#, XDIR#
IR port (IRRX, IRTX, MEDID1)
Ultra I/O
Keyboard/mouse (KBCLK#, KBDAT#, MSCLK#, MSDAT#)
Floppy bus (INDEX#, DIR#, STEP#, WDATA#, WGATE#, TRK0#, WPT#, RDATA#, SLIDE1#, DSKCHG#, MTR0#, MTR1#, DRVSEL0#, DRVSEL1#, DRVDEN0#, DRVDEN1#)
Parallel bus (PD[7:0, SLIN#, INIT#, AFD#, STB#, BUSY, ACK#, PE, SLCT, ERR#)
RS-232 bus (RX, TX, RTS#, CTS#, DTR#, DSR#, DCD#, RI#)
North/south bridge


AGP
Graphics
Controller
AGP 2.0
Pentium
processor
HA[31:0]
HD[63:0]
ADS#
BNR#
BPRI#
DBSY#
DEFER#
DRDY#
HIT#
HITM#
HLOCK#
HREQ[4:0]
HTRDY#
RS[2:0]#
CPURST#
Memory
Controller
Hub (MCH)
DQA[8:0]
DQB[8:0]
RQ[7:5]/ROW[2:0]
RQ[4:0]/COL[2:0]
CTM, CTM#
CFM, CFM#
CMD
SCK
SIO
RDRAM
Hub
interface
Floppy disk
Serial port
Parallel port
Mouse port
K/B port
I/R port
Super I/O
Low Pin
Count (LPC)
interface
I/O Controller
Hub
PCI bus
USB bus
IDE bus
Audio bus (AC'97)
Hub-based


Required pins
Optional pins
Address and data
AD[31:0]
C/BE[3:0]#
PAR
Interface control
FRAME#
TRDY#
IRDY#
DEVSEL#
STOP#
ID SEL
Error reporting
PERR#
SERR#
Arbitration (masters only)
REQ#
GNT#
System
CLK
RST#
AD[63:31]
C/BE[7:4]#
PAR64
REQ64#
ACK64#
64-bit extension
LOCK#
Interface control
INTA#
INTB#
INTC#
INTD#
Interrupts
SBO#
SDONE
Cache support
TDI
TDO
TCK
TMS
TRST#
Jtag

PCI lines


Address1
Data1
Address2
Data2
Address3
Data
PCI bus
(normal mode)
Data3
Address3
Data1
Address1
Data2
Address2
Address
Data1
Data2
Data3
Data4
Data5
PCI bus
(burst mode)
Address
Data1
Data2
Data3
Data4
Data5
Transfer modes

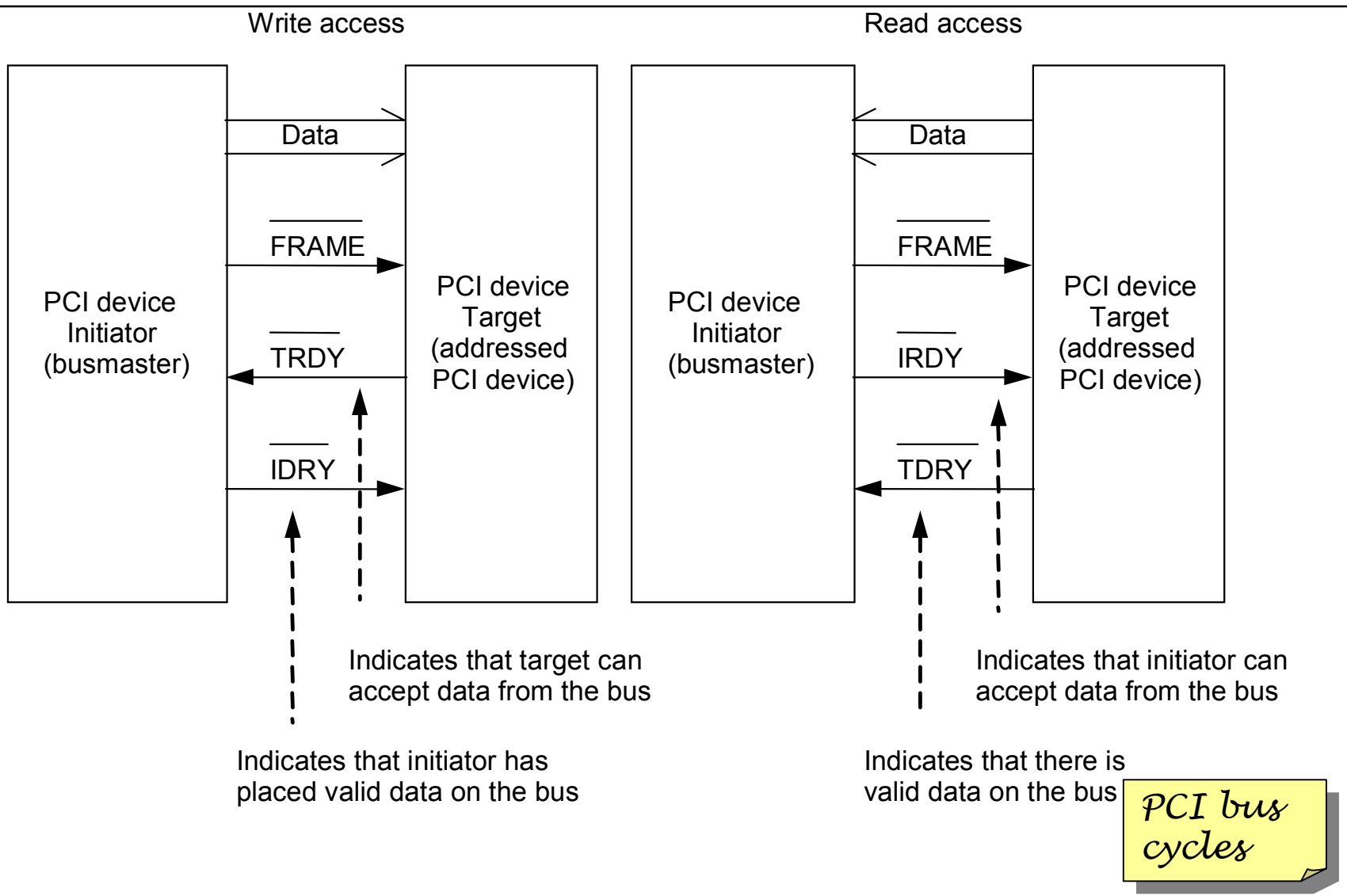

Processor bus
PCI bus
Processor
Address1
Data1
Address2
Data2
Address3
Data3
Address4
Data4
Prefetch buffer
Posting buffer
Prefetch buffer
Posting buffer
Address1
Data1
Data2
Data3
Data4
PCI bridge transfers with burst mode
Primary bus
Secondary bus
Bust mode

C/BE3	C/BE2	C/BE1	C/BE0	*Description*
0	0	0	0	INTA sequence
0	0	0	1	Special cycle
0	0	1	0	I/O read access
0	0	1	1	I/O write access
0	1	1	0	Memory read access
0	1	1	1	Memory write access
1	0	1	0	Configuration read access
1	0	1	1	Configuration write access
1	1	0	0	Memory multiple read access
1	1	0	1	Dual addressing cycle
1	1	1	0	Line memory read access
1	1	1	1	Memory write access with invalidations

For a write operation:

1. **Address phase**. Activate FRAME#. Set C/BE lines. AD31-A0.
2. **Handshaking lines**. Target sets TRDY (Target Ready). Initiator sets IRDY (Initiator Ready).
3. **Data transfer.** Set BE lines to indicate the size of the data transfer.

PCI write transfer



PCI bus cycles

1. **INTA sequence** – addresses an interrupt controller where interrupt vectors are transferred after the command phase.
2. **Special cycle** – used to transfer information to the PCI device about the processor's status. The lower 16 bits contain the information codes, such as 0000h for a processor shutdown, 0001h for a processor halt, 0002h for x86specific code and 0003h to FFFFh for reserved codes. The upper 16 bits (AD31–AD16) indicate x86specific codes when the information code is set to 0002h.
3. **I/O read access** – indicates a read operation for I/O address memory, where the AD lines indicate the I/O address. The address lines AD0 and AD1 are decoded to define whether an 8-bit or 16-bit access is being conducted.
4. **I/O write access** – indicates a write operation to an I/O address memory, where the AD lines indicate the I/O address.
5. **Memory read access** – indicates a direct memory read operation. The byte-enable lines (BE) identify the size of the data access.
6. **Memory write access** – indicates a direct memory write operation. The byte-enable lines (BE) identify the size of the data access.

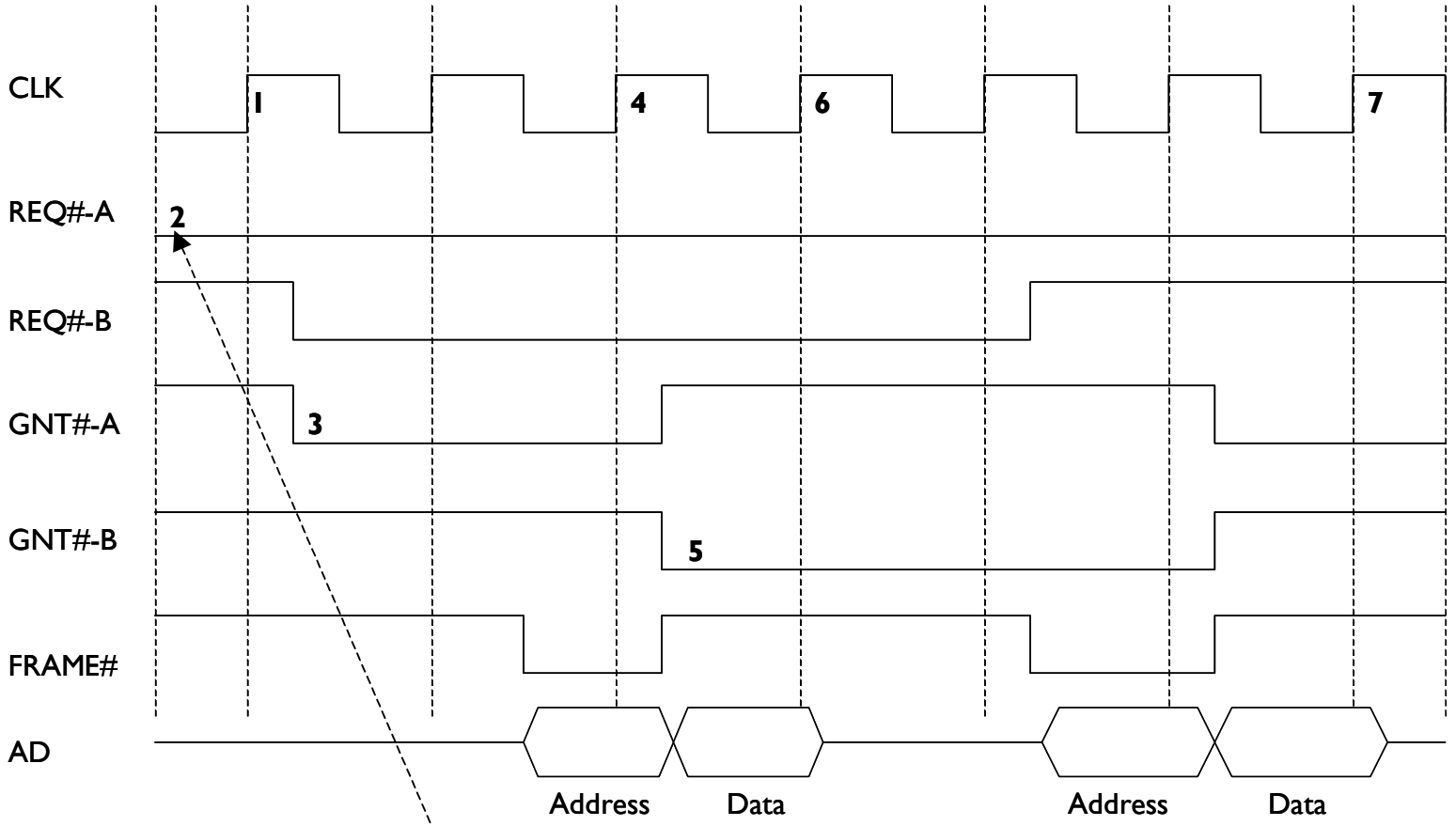
Command lines

1. **Configuration read access** – used when accessing the configuration address area of a PCI unit. The initiator sets the IDSEL line activated to select it. It then uses address bits AD7–AD2 to indicate the addresses of the double words to be read (AD1 and AD0 are set to 0). The address lines AD10–AD18 can be used for selecting the addressed unit in a multi-function unit.
2. **Configuration write access** – similar to the configuration read access, but data is written from the initiator to the target.
3. **Memory multiple read access** – used to perform multiple data read transfers (after the initial addressing phase). Data is transferred until the initiator sets the signal inactive.

Command lines

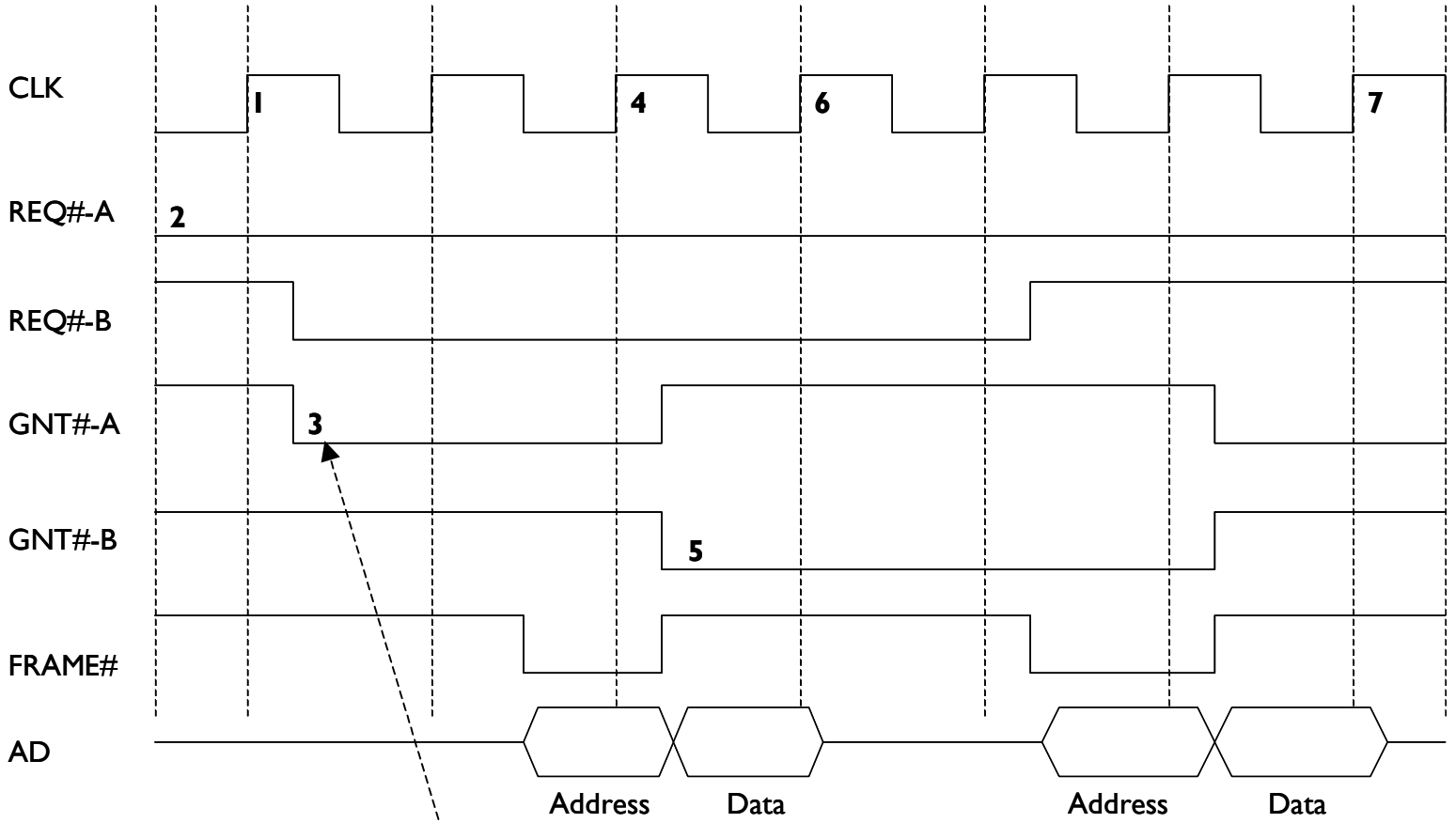
1. **Dual addressing cycle** – used to transfer a 64-bit address to the PCI device (normally only 32-bit addresses are used) in either a single or a double clock cycle. In a single clock cycle the address lines AD63–AD0 contain the 64-bit address (note that the Pentium processor only has a 32-bit address bus, but this mode has been included to support other systems). With a 32-bit address transfer the lower 32 bits are placed on the AD31–AD0 lines, followed by the upper 32 bits on the AD31–AD0 lines.
2. **Line memory read access** – used to perform multiple data read transfers (after the initial addressing phase). Data is transferred until the initiator sets the signal inactive.
3. **Memory write access with invalidations** – used to perform multiple data write transfers (after the initial addressing phase).

Command lines



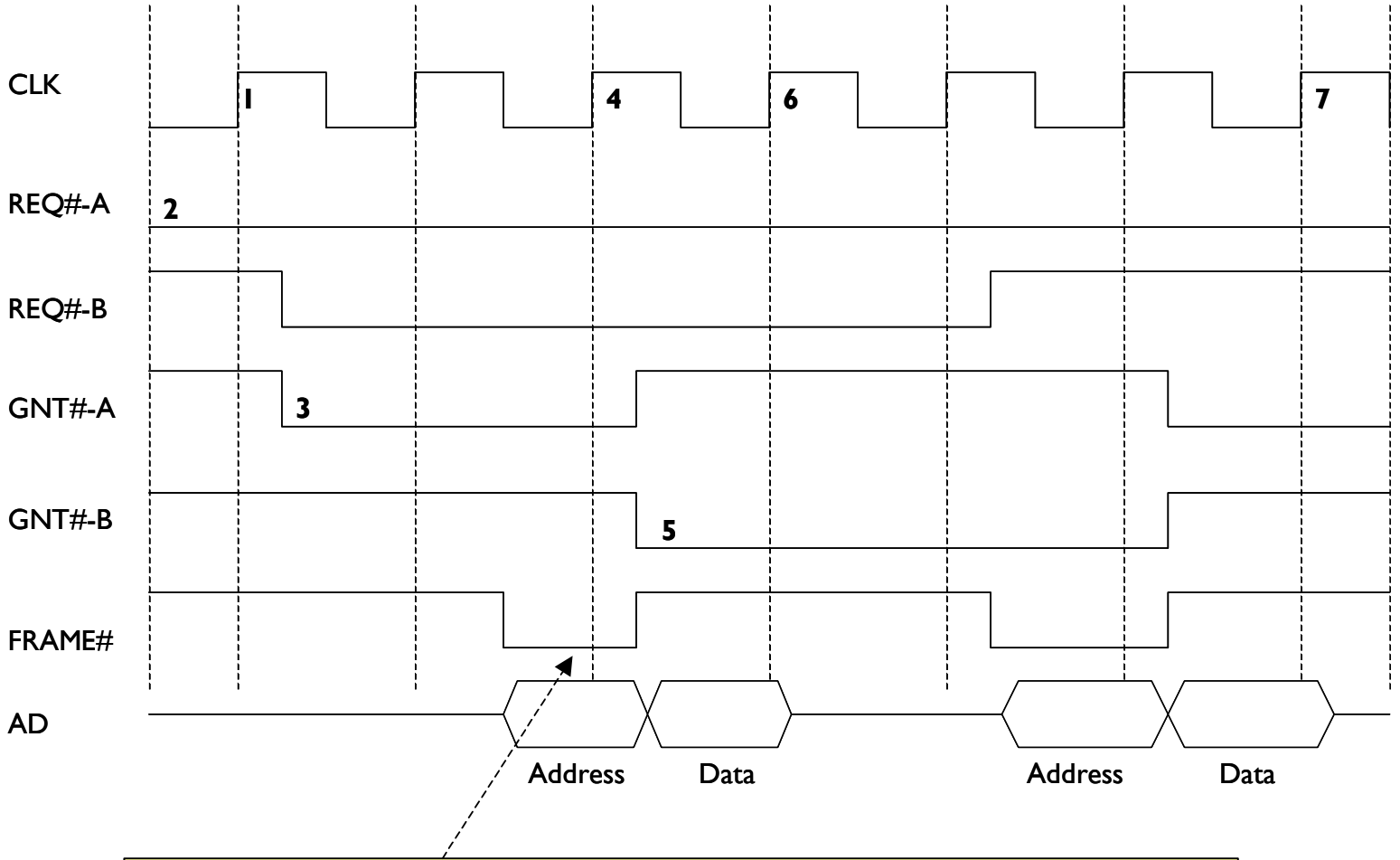
Device-a has asserted REQ#-A line

Bus arbitration



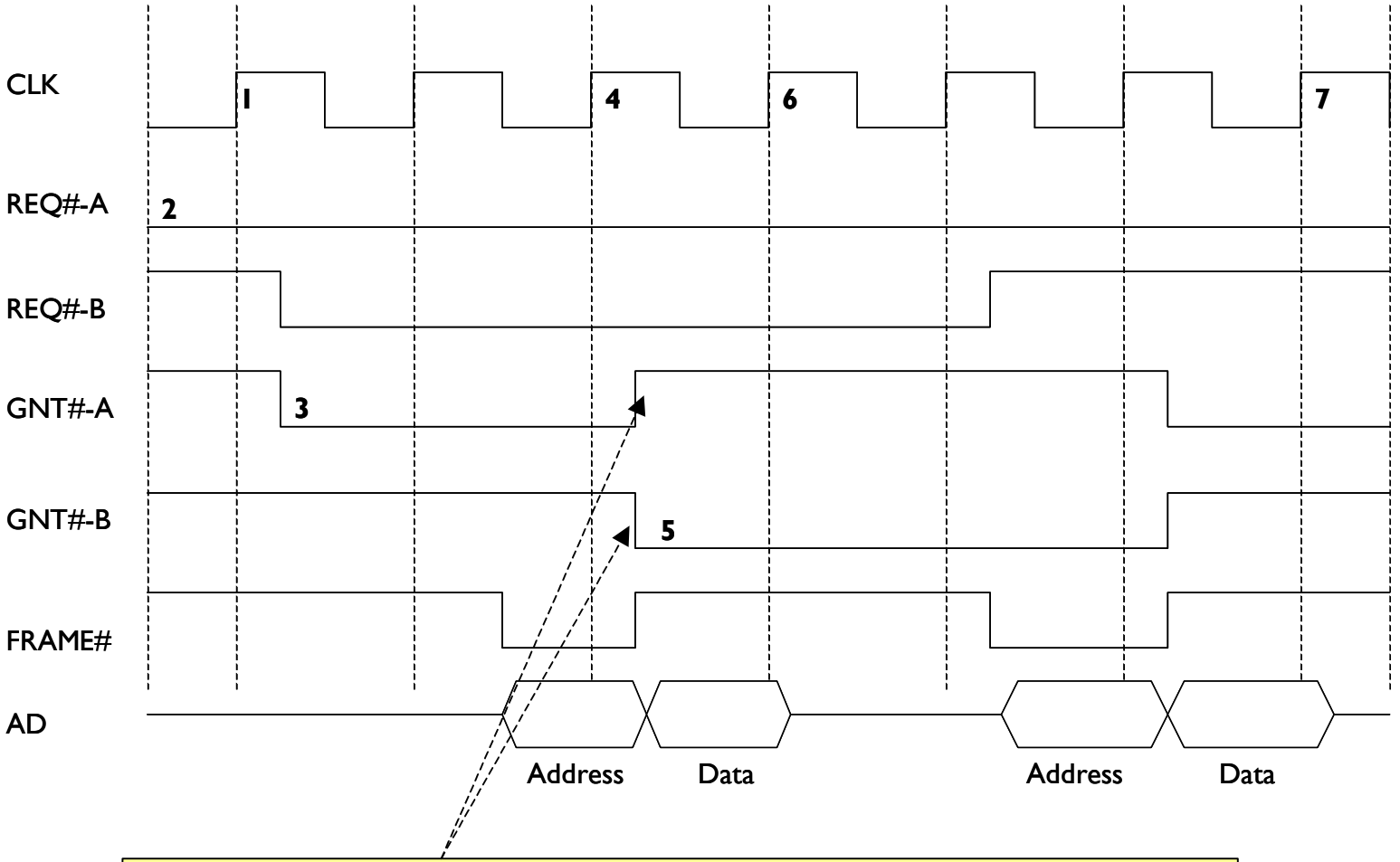
Device-A is granted access to the bus when GNT#-A is asserted.

Bus arbitration

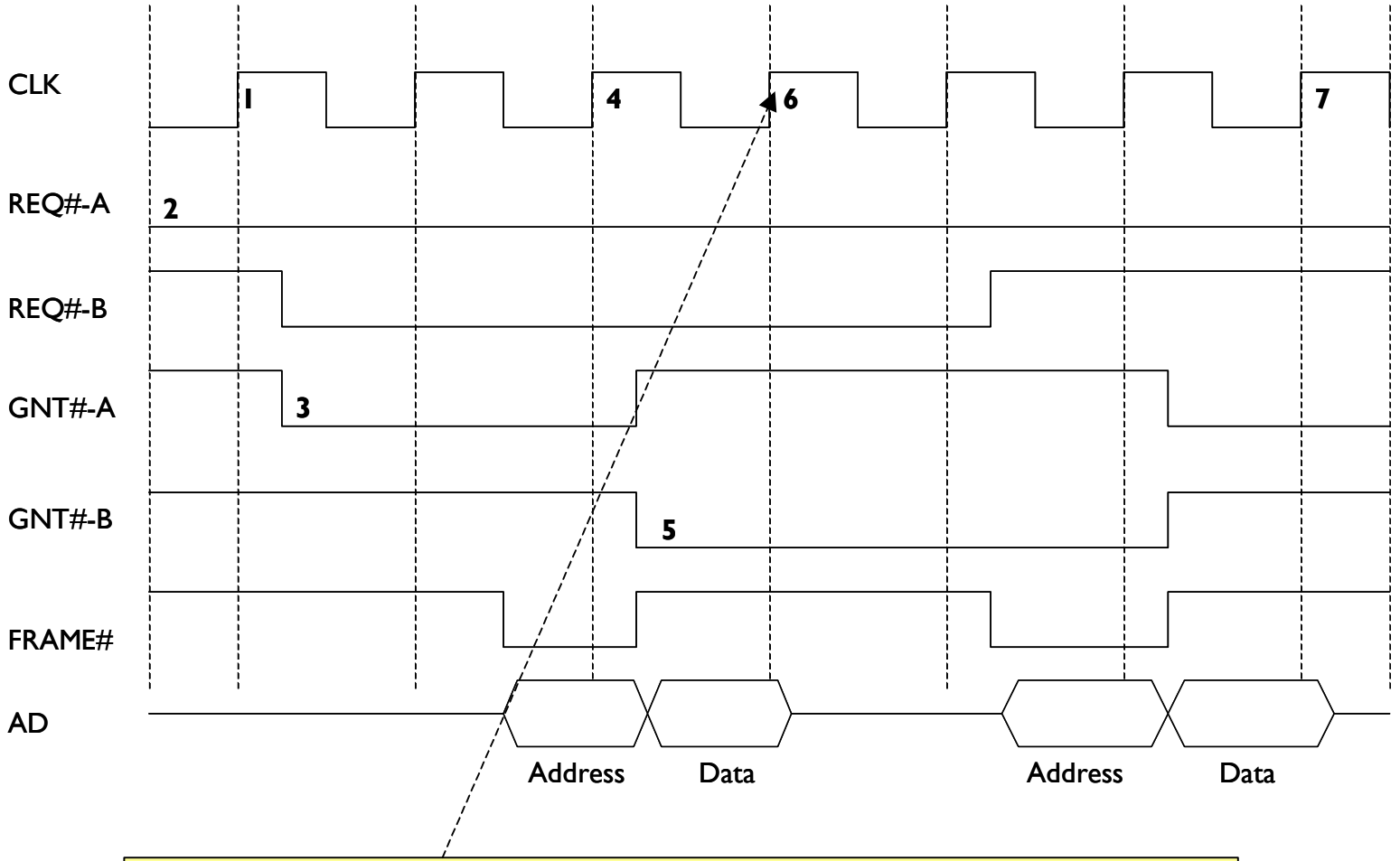


Device-A transaction begins by activating the FRAME# signal

Bus arbitration

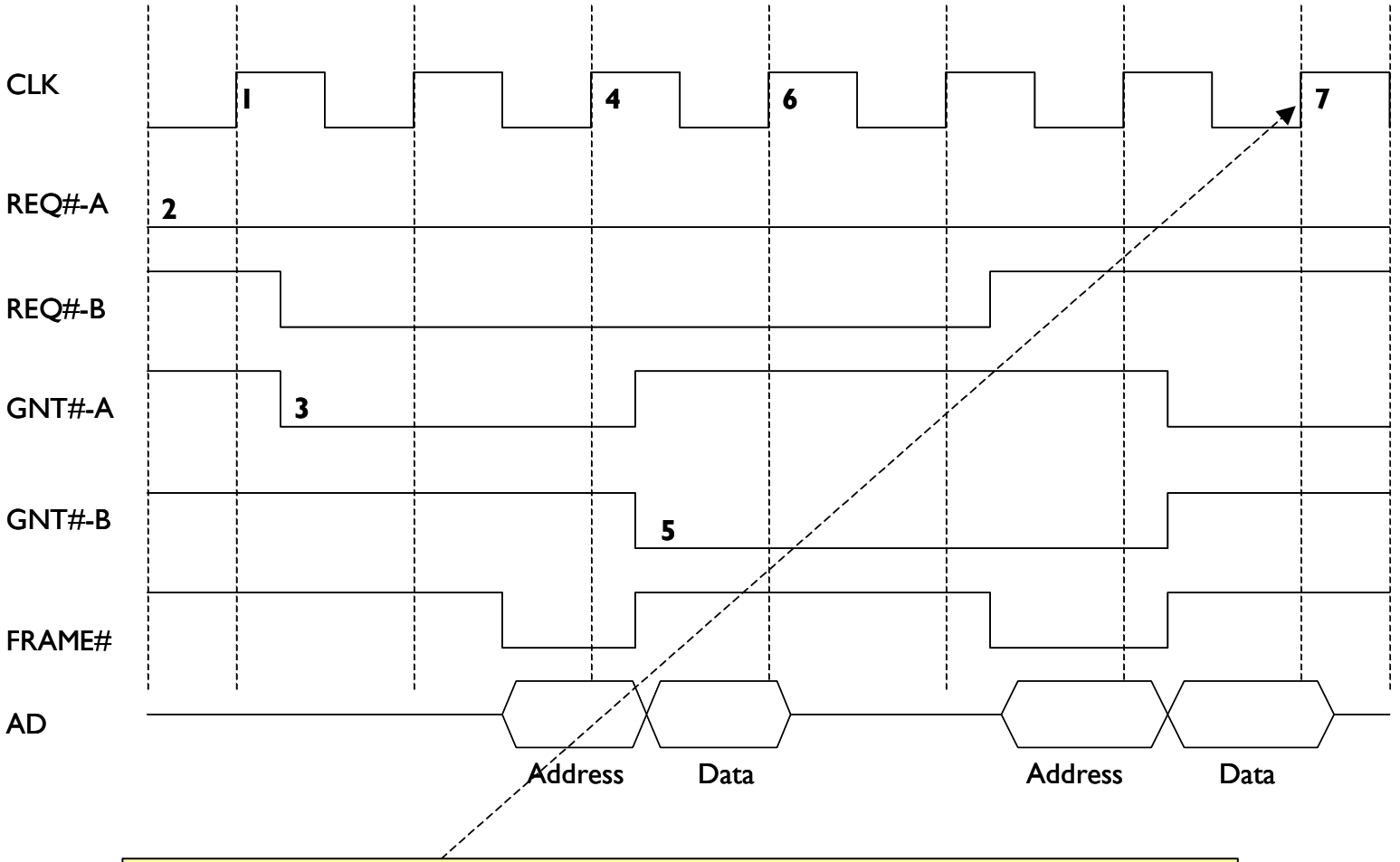


Arbitrator decides that Device-B should get access to the bus next.



Data transferred using TDRY and IRDY. Device-B now owns the bus.

Bus arbitration



Device-B completes its transfer. Arbitrator has allow Device-A access to the bus, as it did not deassert REQ#-A.

PCI allows exclusive access while allowing other masters to access targets other than the locked targets. The sequence is:

- PCI targets that support exclusive accesses sample LOCK# when sampling the address.
- **If LOCK# is asserted during the address phase**, the target of the transaction will **not** mark itself as **locked** (since there is already another exclusive access taking place).
- If LOCK# is **deasserted** during the address phase, then the target of the transaction marks itself as **locked**.
- The target stays locked until FRAME# and LOCK# are deasserted.

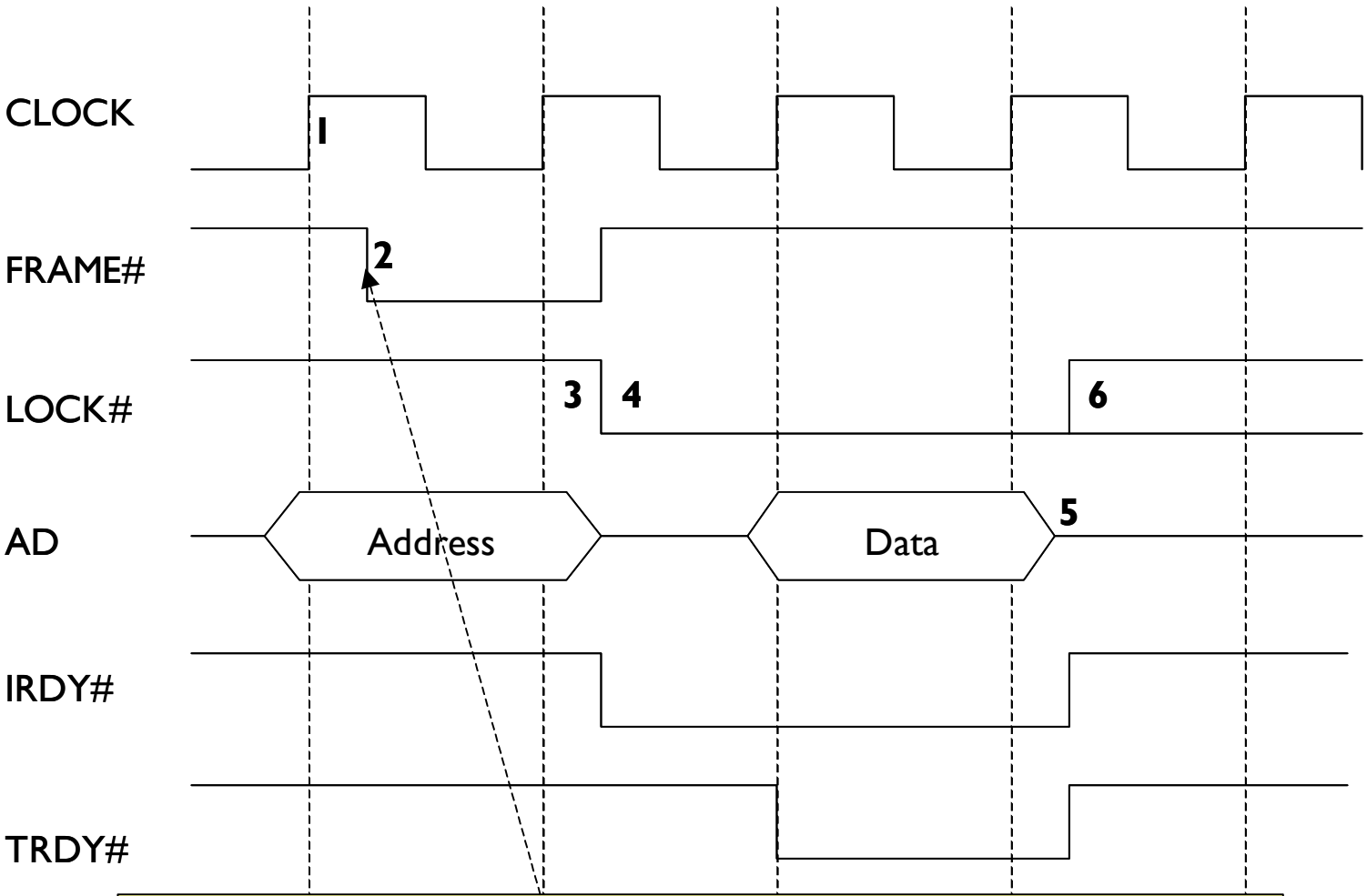
Locked targets

An agent that intends to lock a target must wait until any other locked transactions have ended before requesting and gaining control of the buses.

1. If an agent requires an exclusive operation, then the agent checks the state of the LOCK#.
2. If lock is already asserted, then the agent waits until FRAME#, and LOCK# are de-asserted, (previous lock is released).
3. If LOCK# is not asserted, and FRAME# is not asserted, then the agent asserts its unique REQ# in order to gain control of the buses.
4. When the agent is given control of the buses, (its GNT# is asserted by the arbiter), and LOCK# is not asserted, then the agent, (which is now a master), can assert LOCK#, the clock following the address phase.

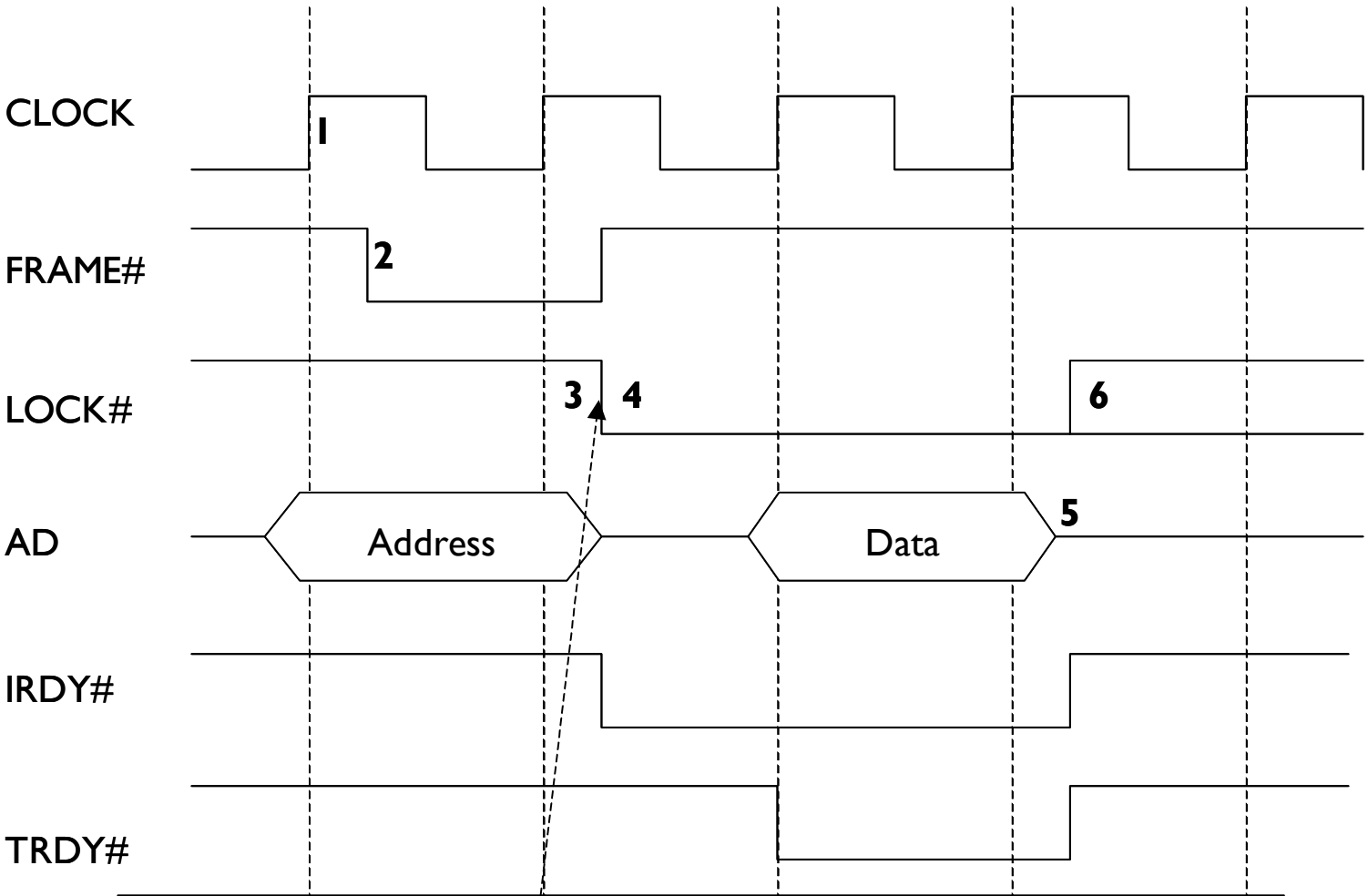
The lock is established after the first data phase is complete.

Establishing a lock



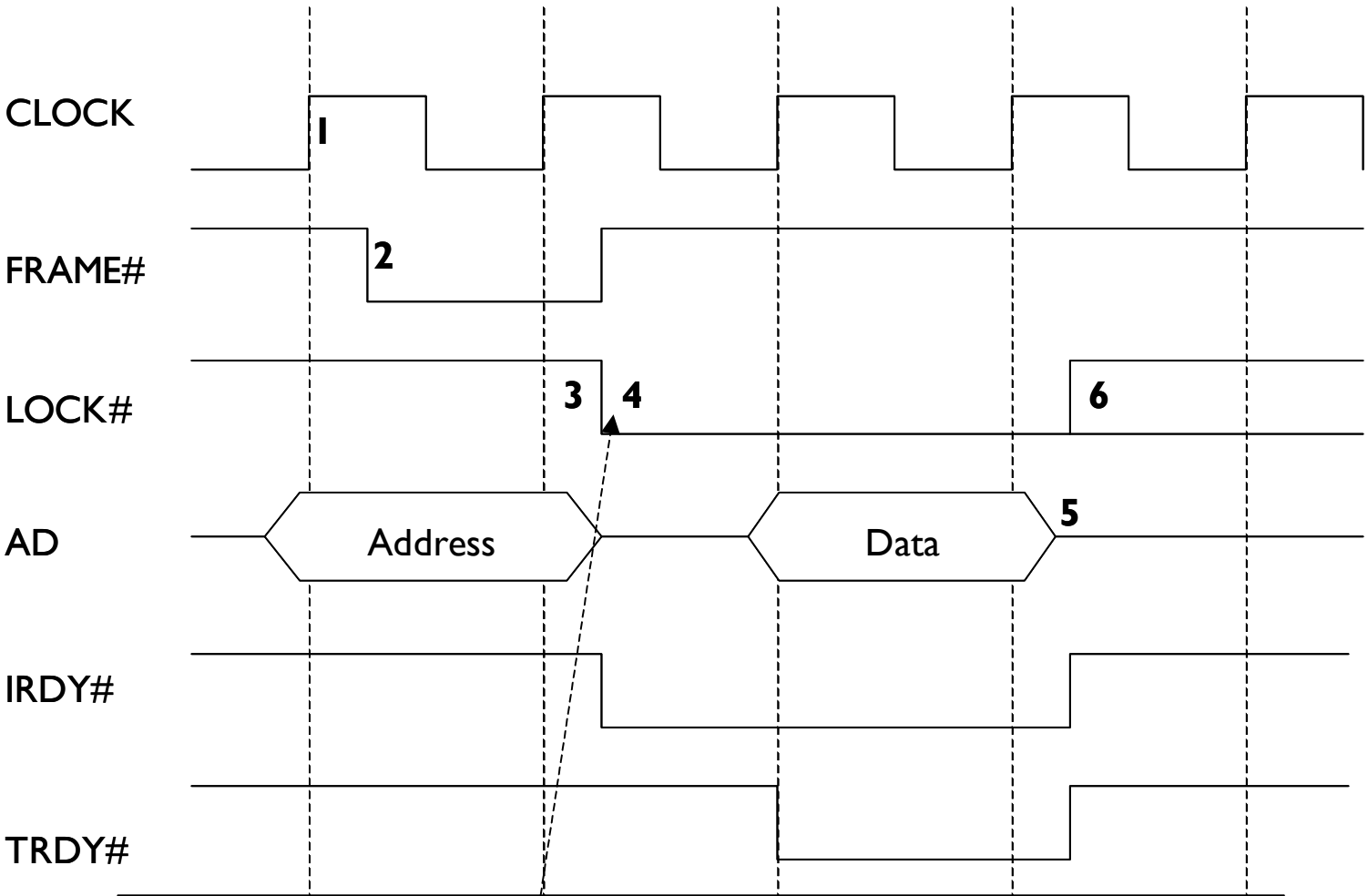
The agent, (master), that requires an exclusive transfer has been granted the buses and begins the transaction by asserting FRAME# and leaving LOCK# deasserted.

OCK timing diagram



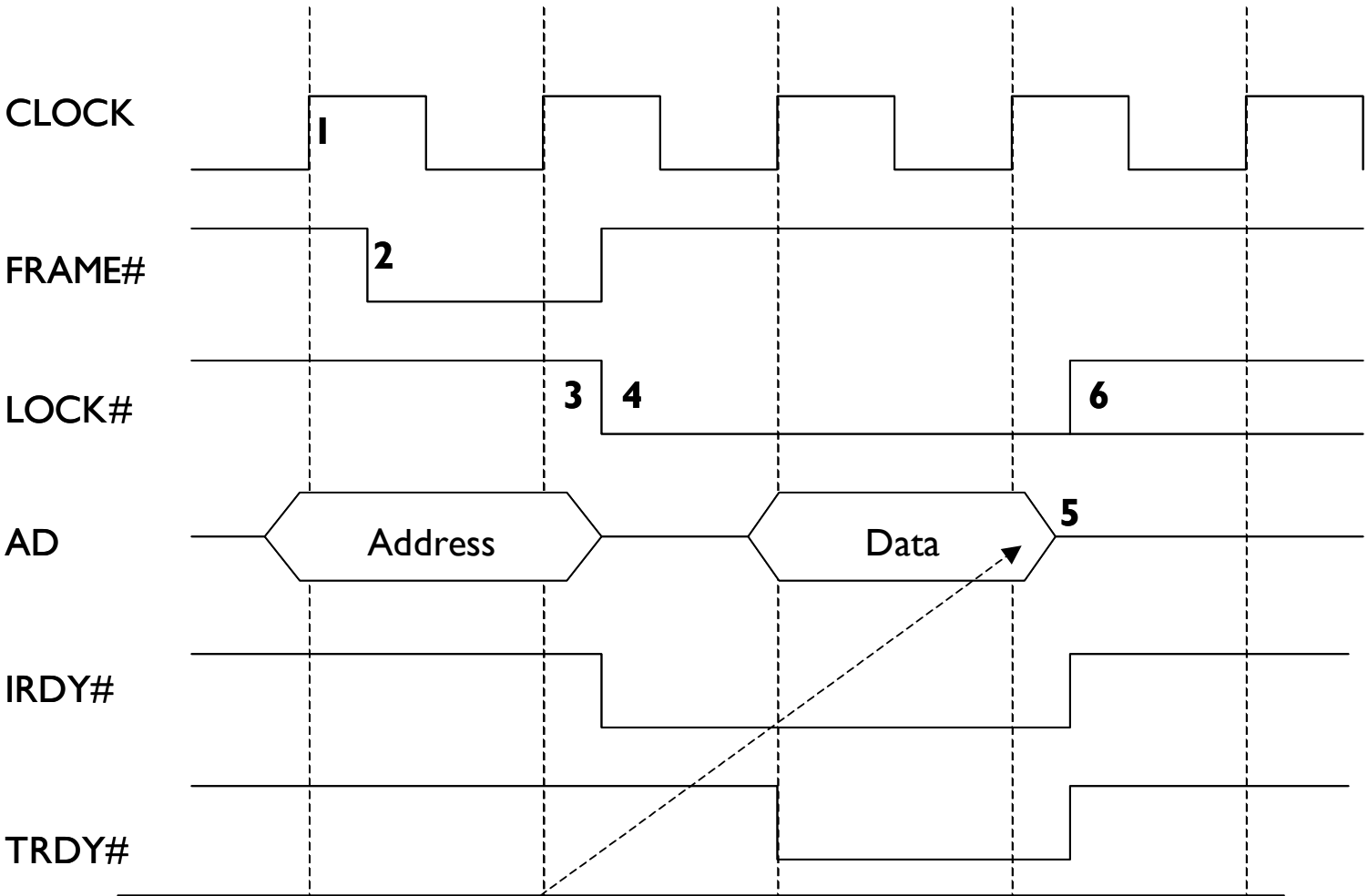
The target samples LOCK# deasserted when address is sampled and marks itself as locked.

OCK timing diagram



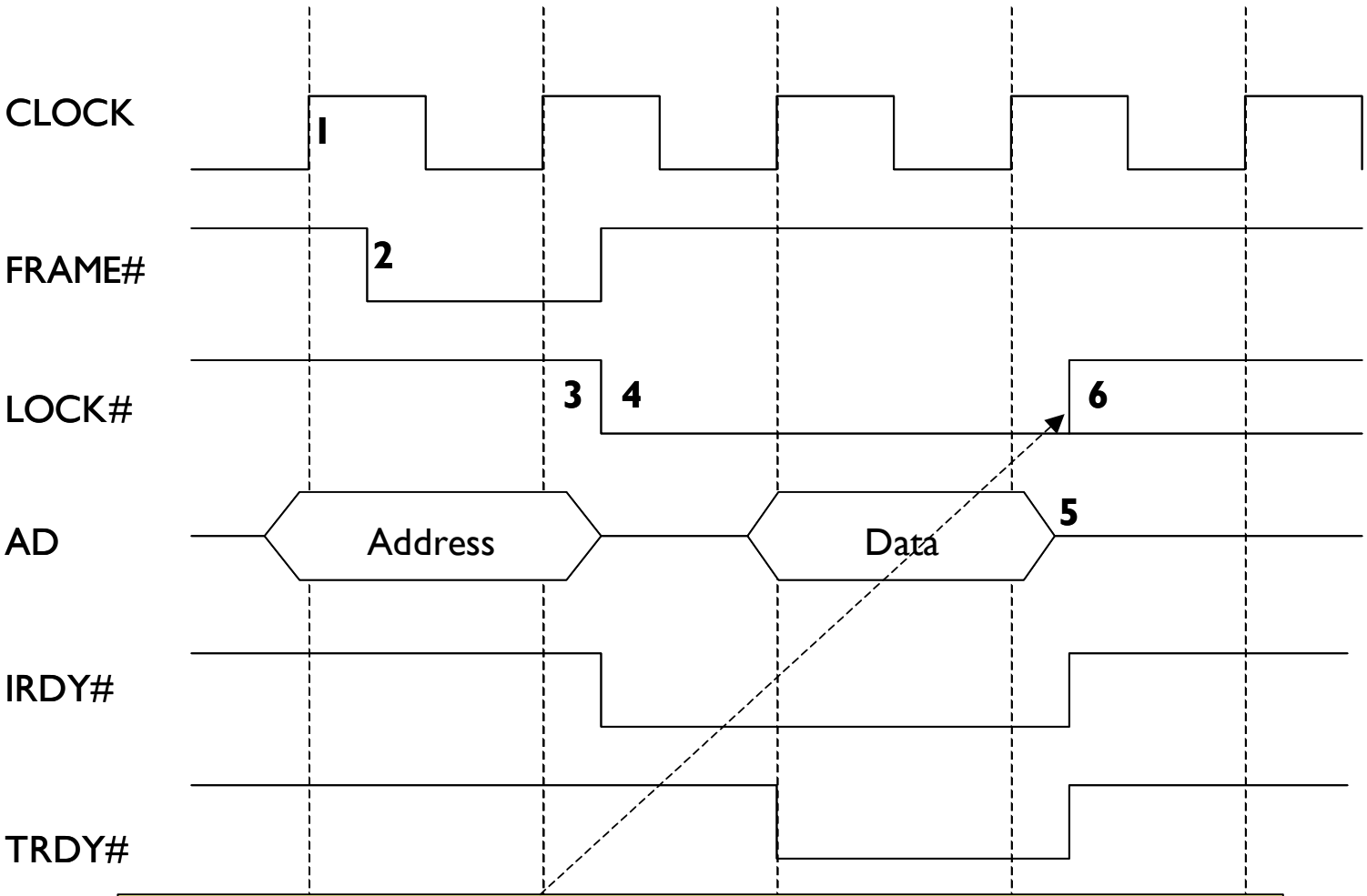
The master asserts LOCK#, the clock following the address phase.

OCK timing diagram



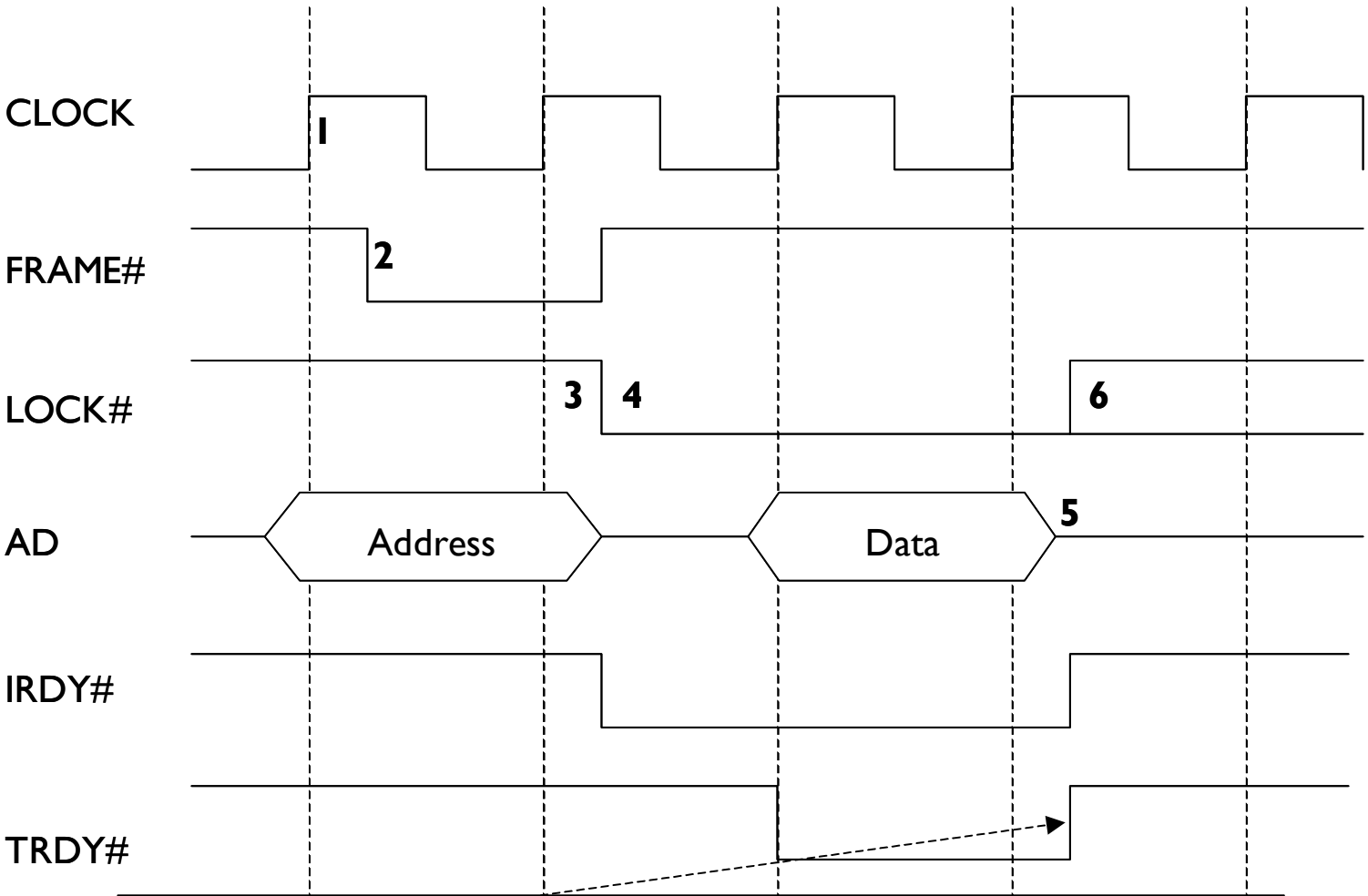
The LOCK# is established after the first data phase is complete.

OCK timing diagram



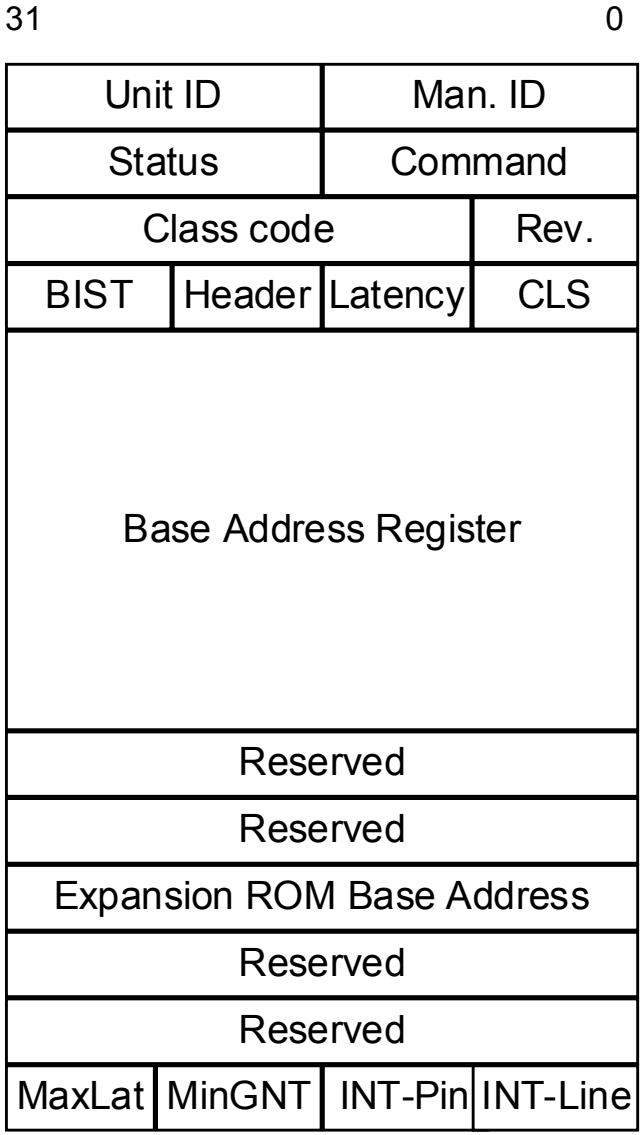
If the master wishes to continue the exclusive access it will continue to drive LOCK# until the LOCK# operation is complete; otherwise, the master will deassert LOCK#.

OCK timing diagram



Indicates that the target (slave) has accepted the data, in response to a write request.

OCK timing diagram



64-byte header in PCI configuration space

- Man ID. Eg Intel is 8086.
- Class code. Eg 0100h identifies a SCSI controller.

Registered used to communicate:

- Configuration address (OCF8h).
- Configuration data (OCFCh)

Configuration space

```
130  Print "Host PCI bridge test"
160  Print "PCI Configuration Address &80000000"
170  IOWRITE &CF8,2,&80000000
180  IOREAD &CFC,2
190  IF B1<>&10008086 THEN GOTO 410
```

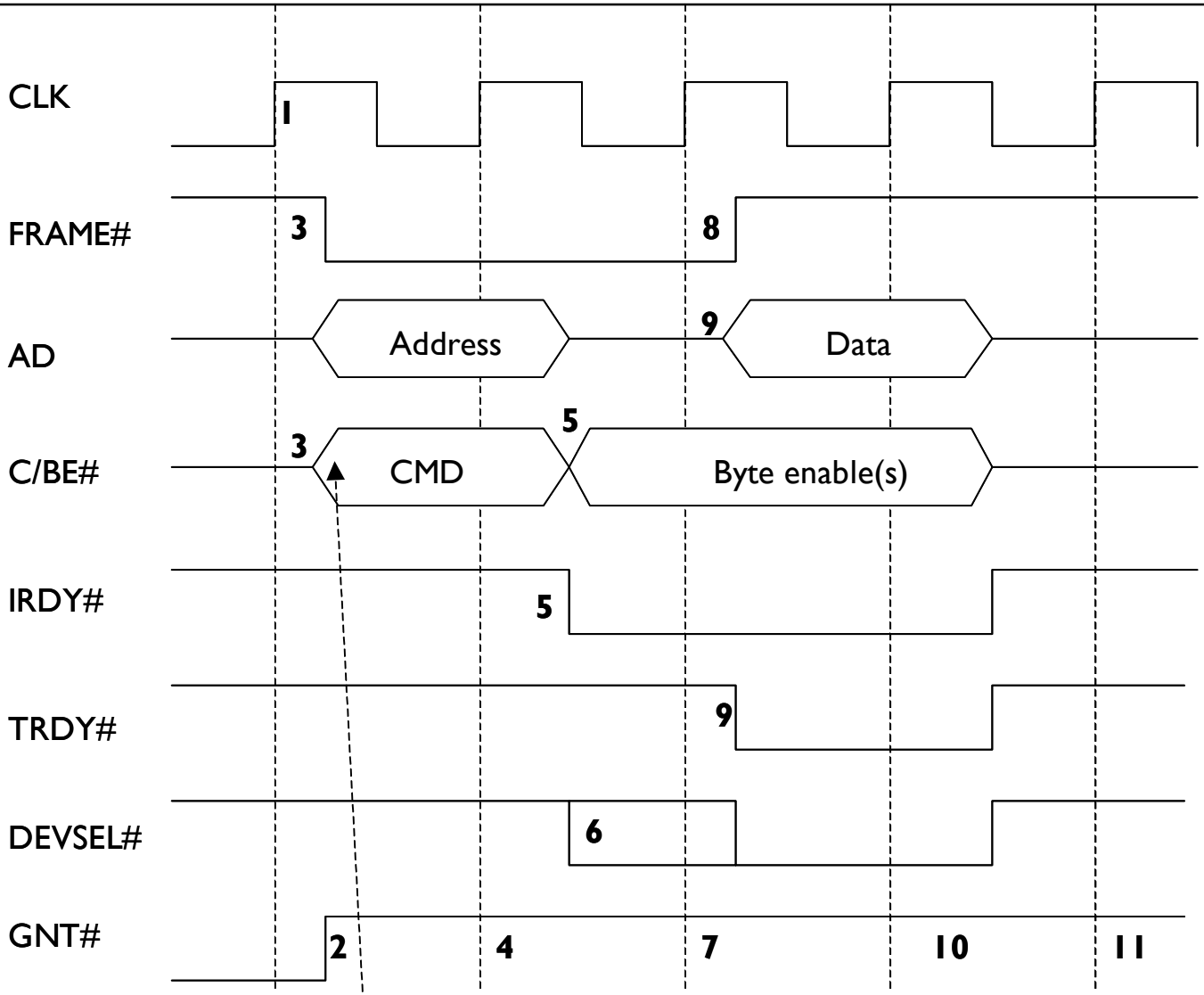
```
210  Dim TEST(4)
220  TEST(1)=&FFFFFFFF
230  TEST(2)=&AAAAAAAA
240  TEST(3)=&O55555555
250  TEST(4)=&O0
260  D9=&80000000
270  REG = &O60
280  REPEAT
290     TST = &O1
300     IOWRITE &CF8,2,D9 + REG
310     REPEAT
320             IOWRITE &CFC,2,TEST(TST)
330             IOREAD &CFC,2
340             If B1 <> TEST(TST) Then GoTo 450
350             TST = TST + &O1
360     UNTIL TST=&5
370     REG = REG + &O4
380  UNTIL REG=&68
```

ration

```
130  Print "Host PCI bridge test"
160  Print "PCI Configuration Address &80000000"
170  IOWRITE &CF8,2,&80000000
180  IOREAD &CFC,2
190  IF B1<>&10008086 THEN GOTO 410
```

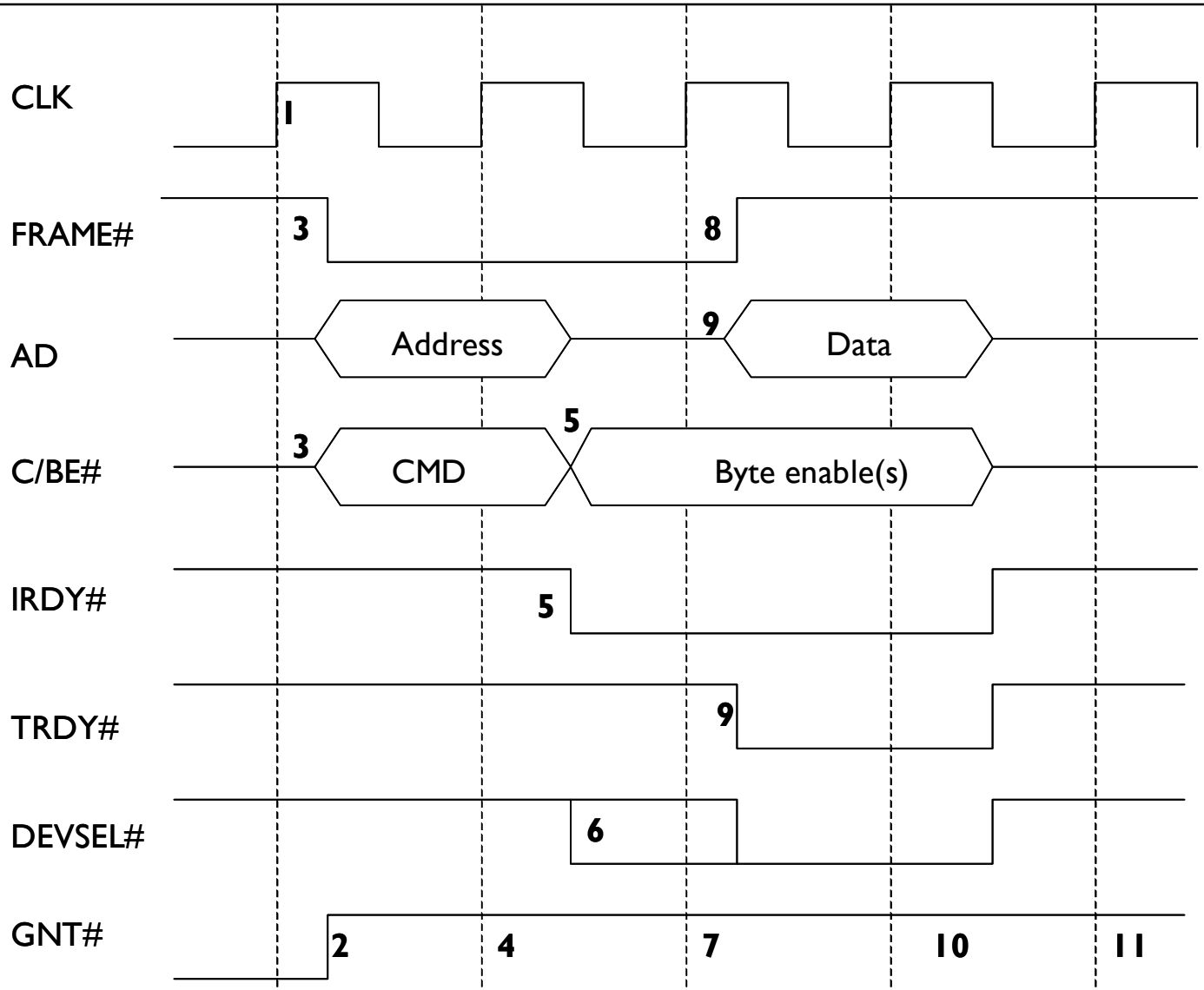
```
210  Dim TEST(4)
220  TEST(1)=&FFFFFFFF
230  TEST(2)=&AAAAAAAA
240  TEST(3)=&O55555555
250  TEST(4)=&O0
260  D9=&80000000
270  REG = &O60
280  REPEAT
290     TST = &O1
300     IOWRITE &CF8,2,D9 + REG
310     REPEAT
320             IOWRITE &CFC,2,TEST(TST)
330             IOREAD &CFC,2
340             If B1 <> TEST(TST) Then GoTo 450
350             TST = TST + &O1
360     UNTIL TST=&5
370     REG = REG + &O4
380  UNTIL REG=&68
```

ration

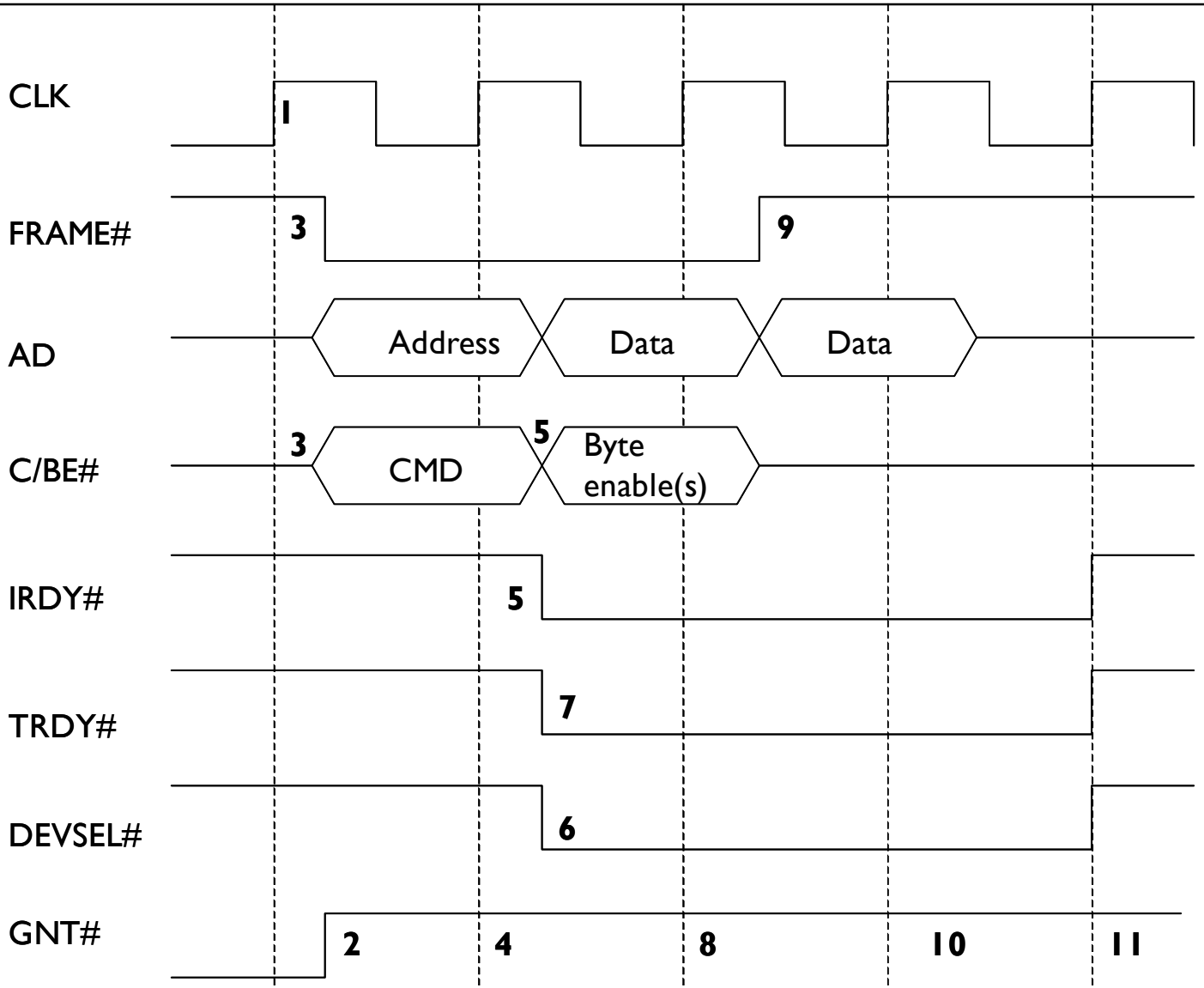


Host performs an I/O write to the configuration address port.

Config register


CLK
FRAME#
AD
C/BE#
IRDY#
TRDY#
DEVSEL#
GNT#
1
3
8
Address
9
Data
3
CMD
5
Byte enable(s)
5
9
6
2
4
7
10
11

Basic read


CLK
FRAME#
AD
C/BE#
IRDY#
TRDY#
DEVSEL#
GNT#
1
3
9
Address
Data
Data
3
CMD
5
Byte
enable(s)
5
7
6
2
4
8
10
11

Basic write