Il bus PCI

Piccinetti Stefano

Prima del bus PCI: il bus ISA

Il bus più diffuso prima del 1992 era il bus ISA (quello sostanzialmente trattato a Reti Logiche).

Il primo bus ISA era ad 8 bit e garantiva una banda limitata.

Una seconda versione, utilizzando i 16 bit, riusciva a garantire una banda di 8,33 MB/s.

Successivamente fu introdotta una estensione di tale bus (EISA) che utilizzava i 32 bit, e garantiva 33 MB/s.

Problemi

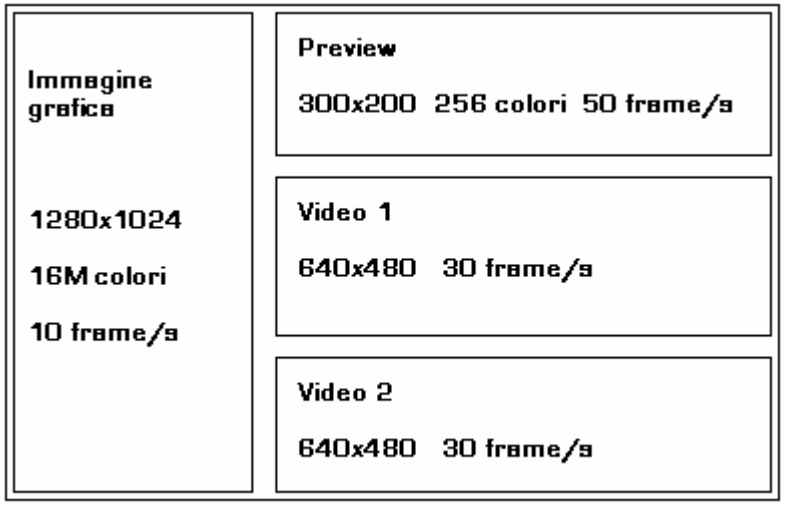
Proprio in quegli anni si andavano sempre più sviluppando applicazioni grafiche.

Tali applicazioni avevano esigenze di banda molto elevate, dunque occorreva fare in modo che una banda sempre maggiore fosse garantita dal bus.

Per fare un semplice esempio, supponiamo di avere una applicazione grafica molto semplice che manda a schermo:

- una immagine
- un'anteprima
- 2 video

Un esempio



Calcoliamo la banda necessaria:

- Immagine => 37,5 MB/s
- Preview => 1,8 MB/s
- Video => 2 x 8,8 MB/s
- Overhead => 0,6 MB/s

TOTALE 57,5 MB/s

E' dunque evidente che col bus ISA/EISA non ci si può fare.

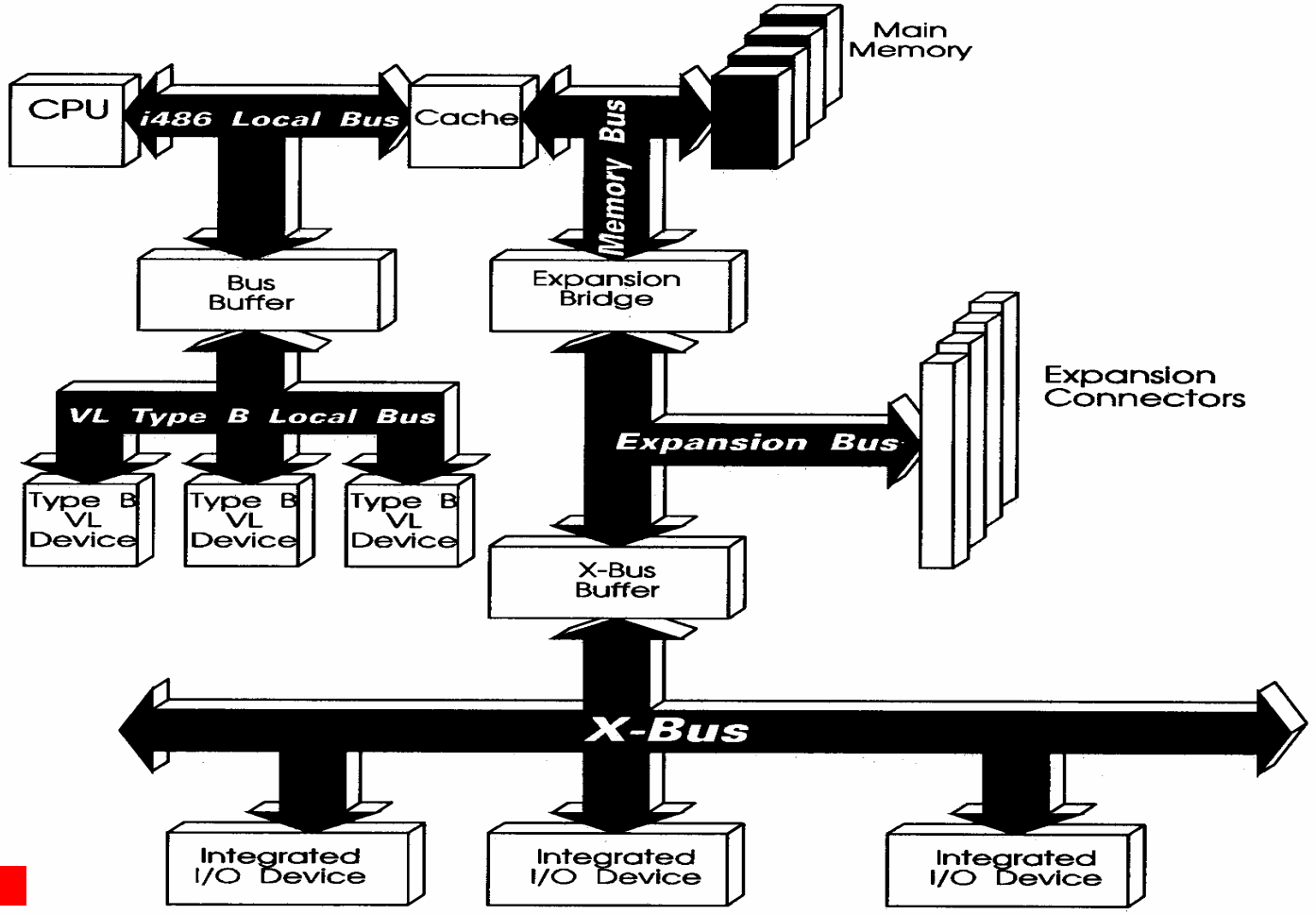
Un primo rimedio

Per rispondere alle esigenze di banda, fu introdotta una nuova architettura del calcolatore, usata nel PC 486.

La novità più importante fu l'introduzione di un Local Bus, ovvero un bus fra CPU e Cache molto veloce, in grado di reggere le velocità delle CPU di allora.

Lo svantaggio era che non si poteva caricare troppo tale bus; al massimo vi si potevano connettere 3 soli dispositivi. Gli altri (quelli più lenti) si ponevano sul vecchio bus ISA.

Architettura del PC 486

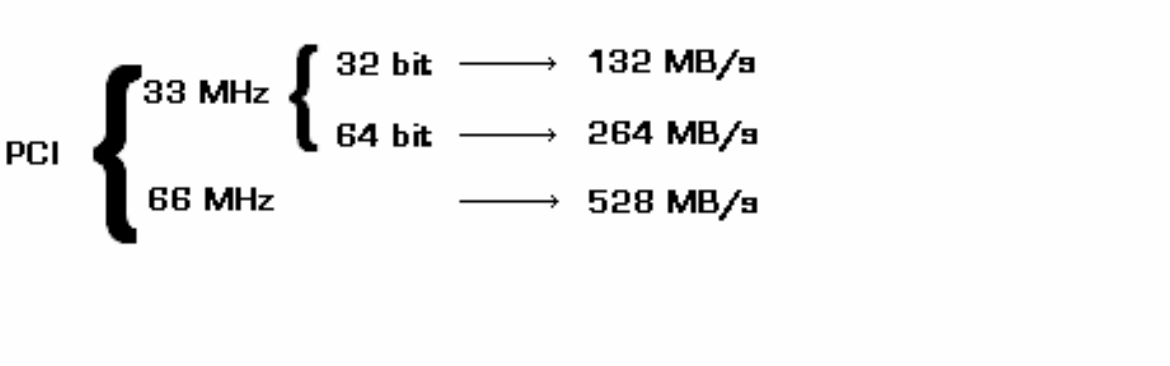


Indietro

Bus PCI

Il bus PCI (Peripheal Component Interconnect) fu introdotto nel 1992 dalla Intel per garantire che sul mercato non ci fosse “affollamento” di varie architetture di bus, inadatte ad un’ottica di lungo periodo.

Oltre a ciò, questo bus è capace di rispondere alle forti esigenze di banda di alcune periferiche e di alcune applicazioni (soprattutto grafiche), offrendo così una soluzione ai problemi precedentemente posti.



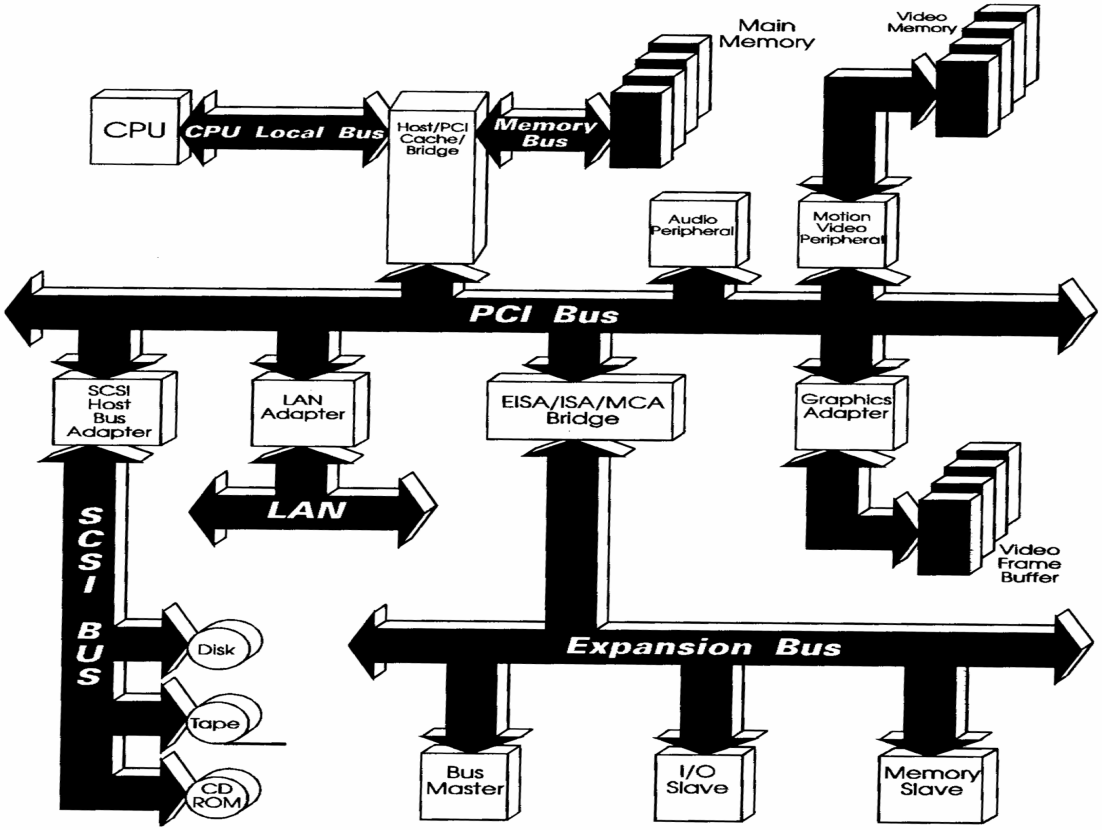
Caratteristiche del bus PCI

Le principali caratteristiche del bus PCI sono:

- Indipendenza dalla CPU
- Possibilità di connettere fino a 256 dispositivi (ogni scheda può implementare diversi dispositivi)
- Basso consumo (questo bus è realizzato con tecnologia c-mos; le linee non sono terminate, dunque si sfruttano le riflessioni)
- Tutti i trasferimenti sono “a burst” (a blocchi)
- Arbitraggio nascosto
- Basso numero di pin (<50)
- 3 spazi di indirizzamento (memoria, I/O e configurazione)
- E’ prevista l’estensione a 64 bit
- Controllo sui trasferimenti (1 bit di parità)

Architettura con bus PCI

Nel 1992, con l'introduzione del bus PCI, l'architettura del PC divenne la seguente:



Considerazioni sulla nuova architettura

- Rispetto all'architettura del PC 486, si è liberato il Local Bus, che ora è dedicato (e più veloce)
- Si mantiene, per ragioni di "compatibilità all'indietro", il bus ISA/EISA
- L'obiettivo di questa architettura è massimizzare il parallelismo sui 3 livelli:
  - Local Bus / Memory Bus
  - PCI bus
  - Expansion bus (ISA/EISA)

Generalità sul protocollo

Le caratteristiche più importanti del protocollo PCI, che è bene tener presente da qui in avanti, sono le seguenti:

- tutti i trasferimenti (sia in lettura che in scrittura) sono a blocchi
- ogni transazione ha 2 fasi:
    1) address phase (di indirizzamento)
    2) data phase (trasferimento dati)
- il protocollo è semisincrono
- l'arbitraggio è nascosto

Dispositivi PCI

Tipici dispositivi PCI consistono in adattatori di periferica completi realizzati su schede elettroniche da montare nei vari slot. Ciascuno slot può realizzare più dispositivi.

Nel bus PCI il montaggio di nuovi dispositivi è molto efficiente, poiché si fa uso del meccanismo del "Plug and Play".

Come in tutti i bus, i dispositivi possono essere Master o Slave. Nel bus PCI i Master sono indicati come **Initiator** e gli Slave come **Target**.

Plug and Play

Il meccanismo del “Plug and Play” consiste nella possibilità di montare dispositivi negli slot del bus facendoli riconoscere alla macchina in fase di bootstrap senza necessità di interventi manuali sulle maschere, che dunque saranno configurate dinamicamente.

Per supportare tale meccanismo si attua una **selezione geografica**; con essa, oltre ad avere le classiche n linee di indirizzi che lavorano con selezione logica, si prevedono tante linee aggiuntive quanti sono gli slot per i dispositivi, e ciascuna di queste linee permetterà di indirizzare direttamente uno slot.

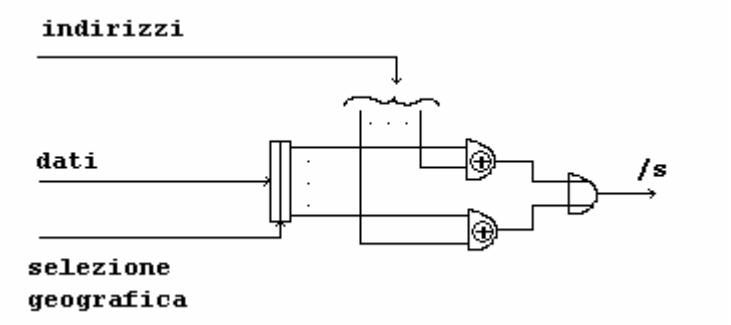
Plug and Play

Indirizzare direttamente uno slot serve, in fase di bootstrap, per andare a scoprire (leggendo in opportuni registri) se in tali slot sono montati dei dispositivi e, se sì, di che tipo di dispositivo si tratta e di che risorse ha bisogno.

Fra le cose da configurare quando si rileva un dispositivo c'è la maschera (che può essere per slot o per dispositivo, poiché una scheda può realizzare più dispositivi).

Per farlo possiamo usare un registro in cui possiamo scrivere l'indirizzo del dispositivo/slot in fase di inizializzazione.

Configurazione dinamica della maschera

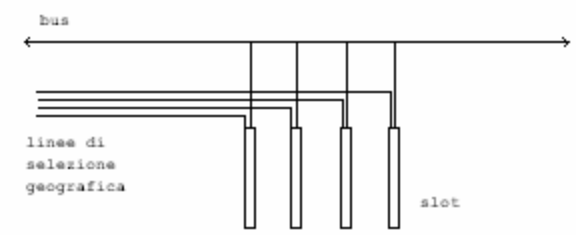


In fase di bootstrap si useranno le linee di selezione geografica, che consentiranno di scrivere nel registro destinato a contenere l'indirizzo dello slot/dispositivo.

In questo registro non si potrà più scrivere, poiché le linee di selezione geografica si usano solo in fase di inizializzazione.

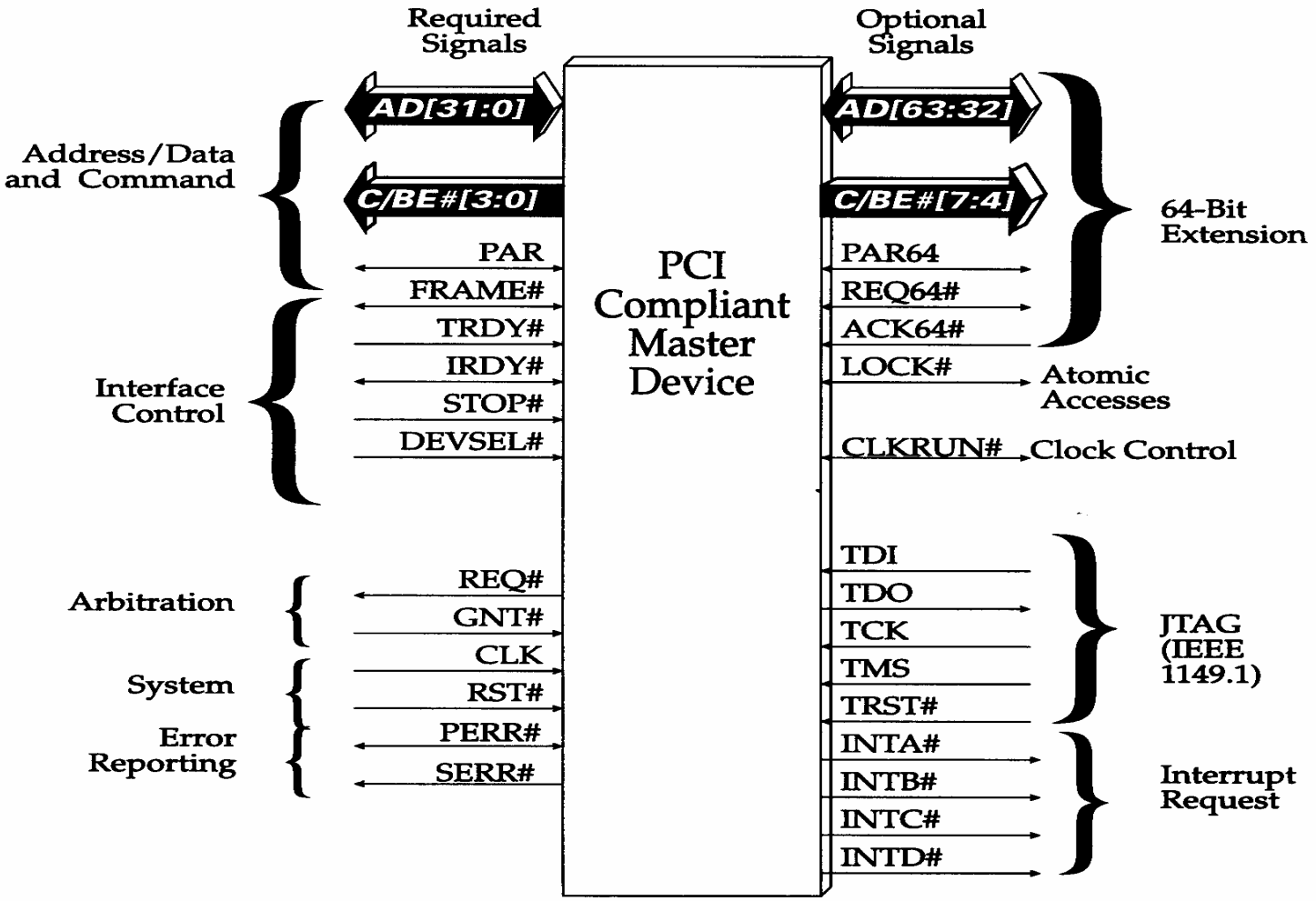
Per poterle utilizzare occorre lavorare nello spazio di configurazione. Nel bus PCI, dunque, si prevedono 3 spazi di indirizzamento:

- memoria
- I/O
- configurazione

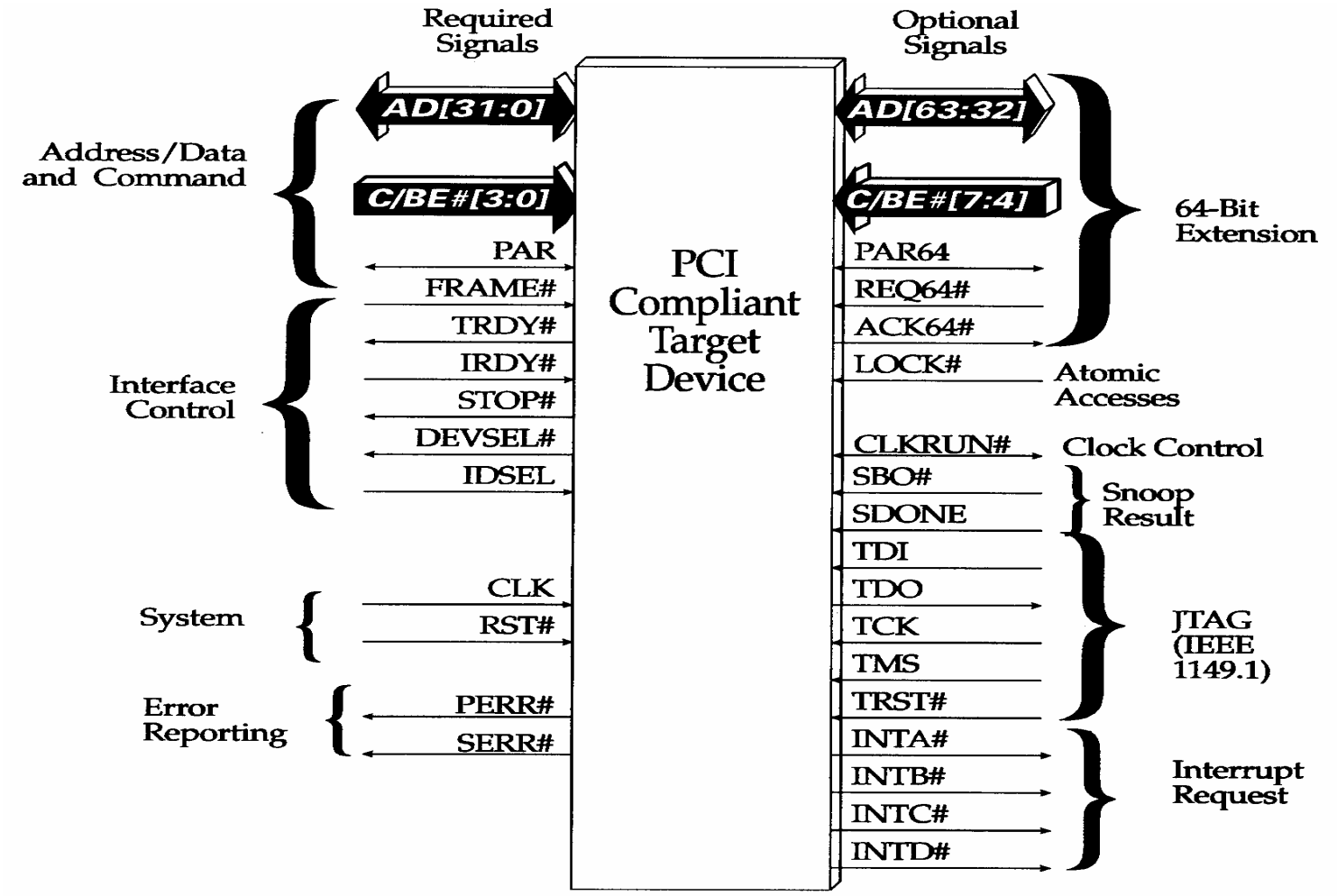


Indietro

Struttura di un Initiator



Struttura di un Target



Linee A/D, C/BE e PAR

Le linee A/D, C/BE e PAR sono linee di informazione a livello, cioè in cui è significativo il loro stato (1 o 0) in un certo istante.

Il bus PCI utilizza:
- 32 linee multiplexate per indirizzi e dati
- 4 linee multiplexate per comandi e bit di bus enable.

=>Ciò consente di ridurre le linee sul bus, e di conseguenza gli effetti di accoppiamento resistivo e capacitivo fra le linee.

Linee A/D, C/BE e PAR

Il protocollo del bus PCI prevede che:

- Nella address phase
  - Le linee A/D sono pilotate da un Initiator e contengono l'indirizzo di un target
  - Le linee C/BE sono pilotate da un Initiator e contengono il comando/tipo dell'operazione
- Nella data phase
  - Le linee A/D, pilotate da Initiator (Write) o Target (Read), contengono dati da trasferire
  - Le linee C/BE, pilotate sempre dall'Initiator, contengono i bit di bus enable

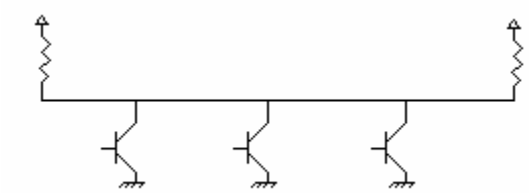
PAR contiene il bit di parità per la rilevazione dell'errore.

Linee per la gestione delle transazioni

Per la gestione delle transazioni si prevedono 6 linee di segnale, cioè linee di cui sono significative le transizioni 0-1/1-0 piuttosto che lo stato ad un certo istante.

Le 6 linee sono:
- /frame
- /irdy
- /trdy
- /devsel
- /lock
- /stop

Le 6 linee sono tutte condivise (fra tutti gli Initiator e tutti i Target), per cui dovranno essere pilotate da porte Open Collector.



Linea /Frame

- Initiator: in / out
- Target: in

Descrizione:

Questa linea è pilotata dall'Initiator corrente (quello che ha vinto l'arbitraggio).
Quando è portata a:
- 0 => indica l'inizio di una transazione
- 1 => indica che l'Initiator è pronto all'ultimo trasferimento dati
(che però non è ancora avvenuto!)

NB: questa linea è anche in ingresso all'Initiator perché a questo, per accedere al bus, non basta avere il /gt, ma deve verificare anche che il bus sia libero (poiché, come si vedrà più avanti, l'arbitro è preemptive).
Il bus è libero quando /frame=1 e /irdy=1.

Linea /Trdy

- Initiator: in
- Target: out

Descrizione:

Il pilotaggio avviene da parte del Target selezionato nella address phase.
Quando è posto a:
- 0 => il Target è pronto ad effettuare il prossimo trasferimento dati
- 1 => il Target comunica all'Initiator che non è pronto ad effettuare il prossimo trasferimento dati; di conseguenza l'Initiator aspetterà il Target

Linea /Irdy

- Initiator: in / out
- Target: in

Descrizione:

Il pilotaggio avviene da parte dell'Initiator.
Quando è posto a:
- 0 => l'Initiator è pronto ad effettuare il prossimo trasferimento dati
- 1 => l'Initiator comunica al Target che non è pronto ad effettuare il prossimo trasferimento dati; quindi il Target aspetterà l'Initiator

- Questa linea è anche in ingresso all'Initiator perché, come /frame, serve per verificare che il bus sia libero.
- Il trasferimento di un dato avviene quando, durante lo stesso ciclo di PCI clock, *entrambe le linee /trdy e /irdy sono a 0.*
- Queste 2 linee (/trdy e /irdy) consentono il protocollo semisincrono.

Linea /Devsel

- Initiator: in
- Target: out

Descrizione:

Questa linea è portata a 0 da un Target quando questo si sente selezionato (ha riconosciuto il suo indirizzo sulle linee A/D durante la address phase).

Se l'Initiator, una volta effettuata la address phase, non vede /devsel a 0 entro 6 cicli di PCI clock, assume che il Target non può rispondere, ed abortisce il trasferimento.

Linea /Lock

- Initiator: in / out
- Target: in

Descrizione:

Questa linea è usata dall'Initiator per bloccare una risorsa su cui vuole fare una transazione atomica (indivisibile).

Come chiarito meglio più avanti, il meccanismo del locking nel bus PCI è efficiente, poiché evita il bloccaggio di tutto il bus, limitandosi a bloccare solo la risorsa interessata alla transazione.

Linea /Stop

- Initiator: in
- Target: out

Descrizione:

Un Target che risulta bloccato tramite il meccanismo del locking, deve poter inibire ogni tentativo di transazione da parte di Initiator diversi dall'Initiator bloccante.

La linea /stop serve appunto per comunicare ad un Initiator (che non è il bloccante) che tenta di effettuare una transazione che il dispositivo è bloccato.

Altre linee

Altre linee che compaiono nella schematizzazione di Initiator e Target sono:

- /req e /gnt: linee per l'arbitraggio
- clk e /rst: linee per il PCI clock e per il reset asincrono
- /perr e /serr: linee per gli errori
- altre linee per l'estensione ai 64 bit

Il clock PCI costituisce un input per tutti i dispositivi che risiedono sul bus PCI, arbitro incluso.

Transazioni PCI

Ogni transazione PCI è composta di 2 fasi:

- Address phase
- Data phase(s)

Address Phase

Ogni transazione PCI inizia con una address phase della durata di 1 ciclo di PCI clock.

Durante questa fase:

- l'Initiator identifica il dispositivo target ed il tipo di transazione; pone il segnale /frame a 0 ad indicare che sul bus sono presenti indirizzo e comandi validi
- Dal momento in cui /frame=0, è compito di ogni target PCI riconoscere eventualmente il proprio indirizzo
- Quando un target si sente selezionato, risponde all'Initiator ponendo /devsel=0

Data Phase(s)

Finita la address phase, si passa alla data phase, durante la quale si ha trasferimento dati fra Initiator e Target.

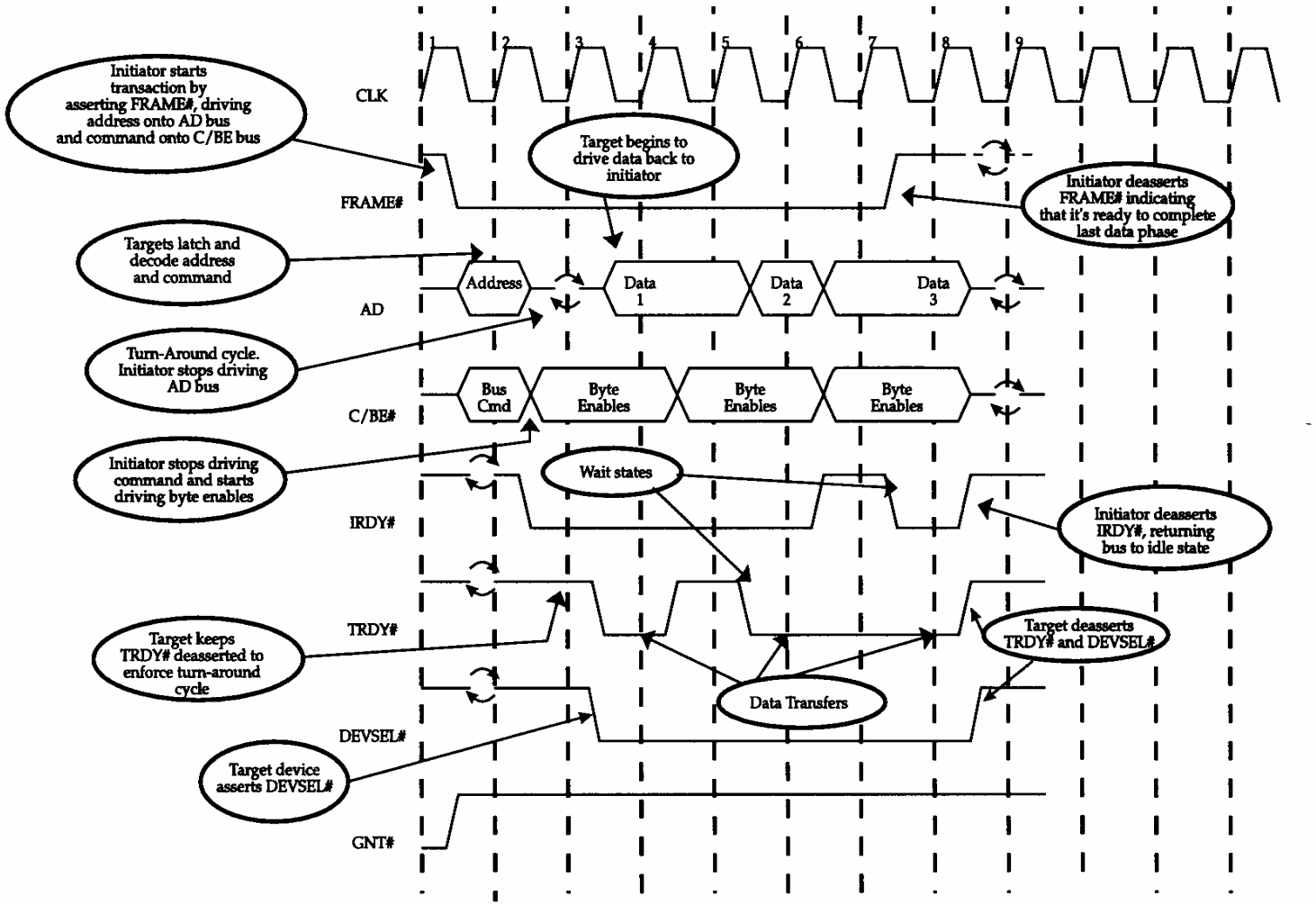
Il numero dei trasferimenti da effettuare è variabile; l'Initiator indica la durata di una transazione ponendo:
- /frame=0 all'inizio della address phase
- /frame=1 all'inizio dell'ultimo trasferimento (da ciò il Target si accorge quando termina la data phase)

I byte delle linee A/D coinvolti nel trasferimento sono quelli che hanno il relativo bit delle linee C/BE a 0.

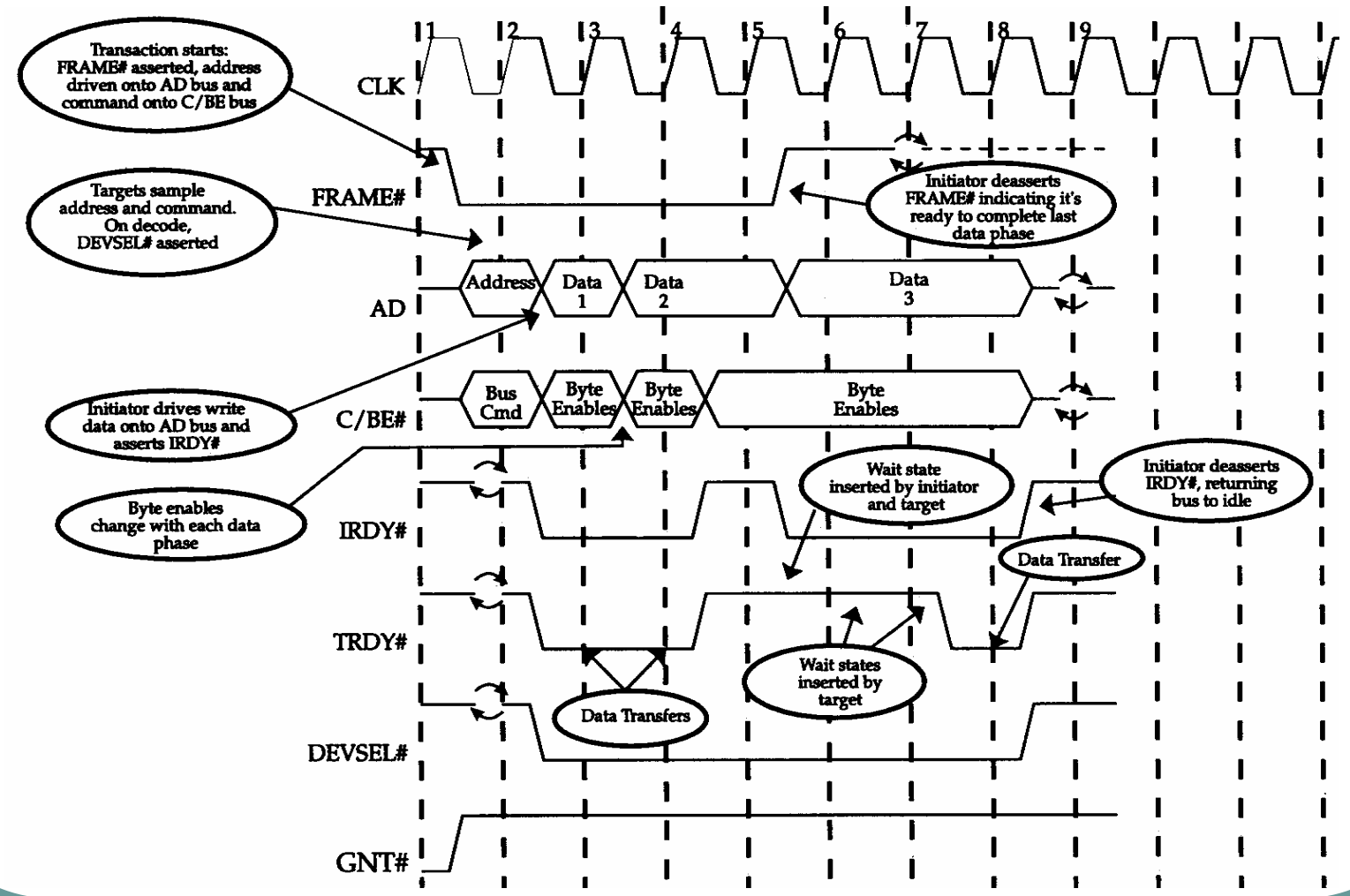
Sia Initiator che Target devono comunicare (tramite /irdy e /trdy) che sono pronti al prossimo trasferimento dati, altrimenti si prolunga il trasferimento finché entrambi non sono pronti.

Il trasferimento avviene quando, nello stesso ciclo di PCI clock, /trdy=0 e /irdy =0.

Operazione di Read



Operazione di Write



Arbitraggio

In un certo istante, 1 o più Master PCI potrebbero aver bisogno del bus per effettuare dei trasferimenti di dati con altri dispositivi.

=> C'è dunque bisogno di un arbitraggio

Ogni Master è connesso con l'arbitro tramite i classici collegamenti /req e /gnt: quando un Master ha bisogno del bus, informa l'arbitro della sua richiesta ponendo /req=0.

Arbitraggio nascosto

Il bus PCI presenta un arbitraggio un po' particolare.

"*Arbitraggio nascosto*" significa che l'arbitraggio ha luogo mentre un Initiator sta utilizzando il bus per una transazione.

Ciò significa che l'arbitro può decidere di assegnare il grant ad un altro Initiator senza che l'ultima transazione sia terminata: **l'arbitraggio è dunque preemptive**.

Bus libero

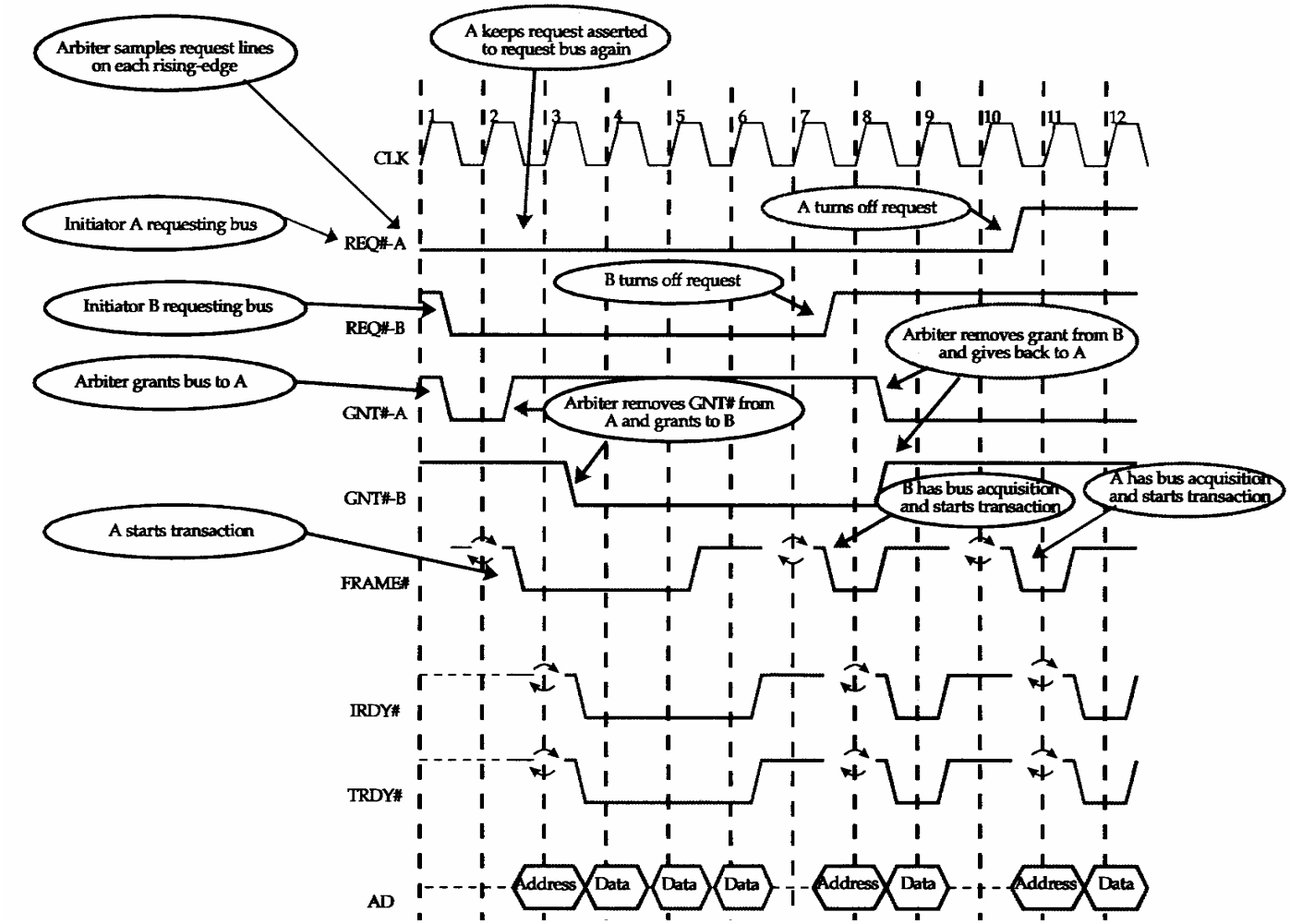
L'Initiator che riceve il /gnt non può però utilizzare subito il bus.

Per utilizzarlo, infatti, occorre che un Initiator:

- abbia il /gnt
- veda il bus libero (/frame=1, /irdy=1)

Dunque non ci sarà conflitto se assegno il /gnt ad un altro Initiator mentre la transazione del precedente non è terminata.
L'Initiator che riceverà il /gnt, infatti, vedrà il bus occupato.

Temporizzazione arbitraggio



Perché un arbitraggio nascosto?

L'arbitraggio nascosto realizzato nel bus PCI permette una maggiore efficienza dello stesso.

Infatti:

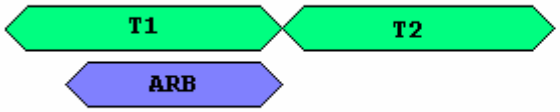
- con un arbitraggio **non nascosto** 2 transazioni sono divise da una fase di arbitraggio, durante la quale si stabilisce quale sarà il prossimo Initiator ad usare il bus.
  In questo caso la fase di arbitraggio è puro overhead, poiché durante essa il bus rimane inutilizzato.
- con un arbitraggio **nascosto**, invece, si stabilisce quale sarà il prossimo Initiator a prendere possesso del bus mentre la transazione corrente è ancora in corso.
  Quindi, quando quest'ultima terminerà, l'Initiator che sarà stato selezionato precedentemente (con /gnt=0) accederà subito al bus (perché lo vedrà libero), il quale rimarrà inutilizzato per un tempo minimo.

Arbitraggio nascosto e non: confronto

Consideriamo 2 transazioni (T1 e T2) ed una fase di arbitraggio (ARB).

Arbitraggio non nascosto

Arbitraggio nascosto



Si ottimizza perché transazioni ed arbitraggio avvengono in parallelo.

Locking

Problema: A volte un Initiator ha bisogno di più accessi consecutivi al bus.

Esempio: INC a.

Occorrono 2 accessi al bus: uno in lettura ed uno in scrittura.
Questi accessi, inoltre, devono essere **consecutivi.**

Infatti, se così non fosse, potrebbe succedere che, dopo il primo accesso, un altro Initiator vinca l'arbitraggio e vada a leggere a, trovandolo in uno stato non consistente.

Occorrerà dunque poter garantire accessi al bus atomici, cioè indivisibili.
Per fare ciò, si utilizza il meccanismo del "locking".

Possibili soluzioni

Una prima soluzione consiste nel dotare gli Initiator di un piedino “lock”, che andrà in ingresso ad un circuito che, quando lock=1, maschererà all’arbitro il rilascio della richiesta da parte dell’Initiator che intende fare operazioni atomiche.

Dal punto di vista dell’arbitro, è come se tale Initiator non rilasciasse la richiesta finché non ha terminato tutte le sue operazioni.

Svantaggi:

Una tale soluzione è pesante, poiché comporta il blocco di tutto il bus finché le operazioni da eseguire in maniera atomica non sono tutte terminate.

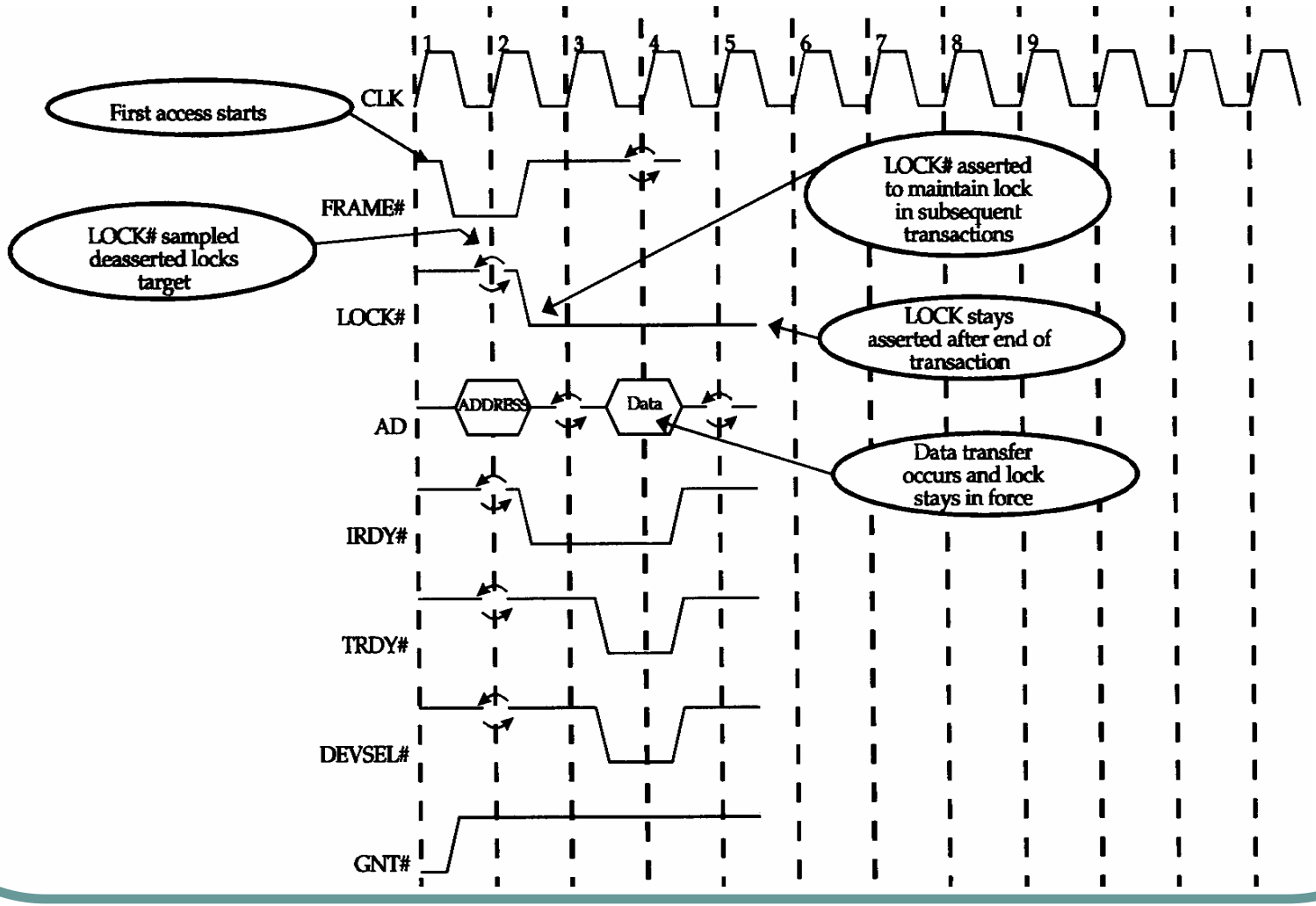
Soluzione PCI

Nel bus PCI si adotta una soluzione migliore.
Non si ricorre al blocco di tutto il bus, ma solo della risorsa (Target) su cui si vogliono effettuare operazioni atomiche.

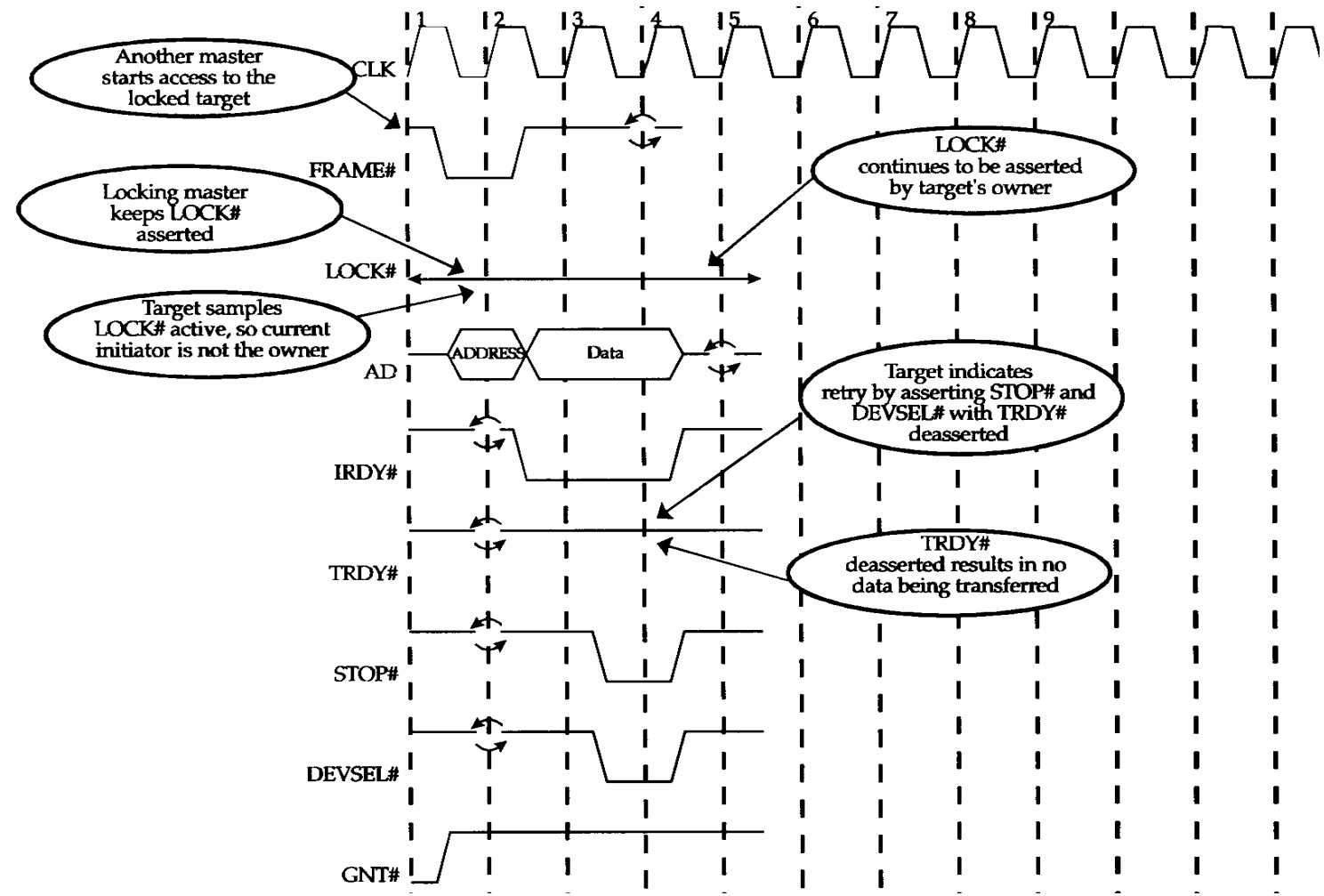
Specifiche generali:

- Si prevede 1 sola linea /lock, condivisa da tutti i dispositivi e quindi pilotata da porte Open Collector
- Si ha la possibilità di bloccare 1 solo dispositivo (Target) alla volta
- Se un Initiator intende bloccare un Target, dovrà porre la linea /lock a 0 al PCI clock successivo la address phase (solo però se prima si trovava a 1, cioè se non ci sono altri dispositivi bloccati)
- Un Target bloccato potrà respingere le richieste di altri Initiator tramite la linea /stop

Bloccaggio di una risorsa



Respinta di un Initiator



Riconoscimento dell'Initiator bloccante