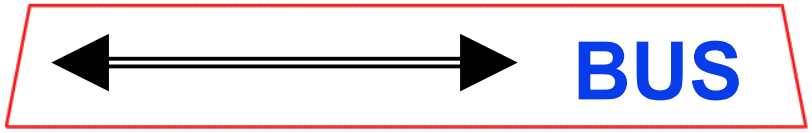
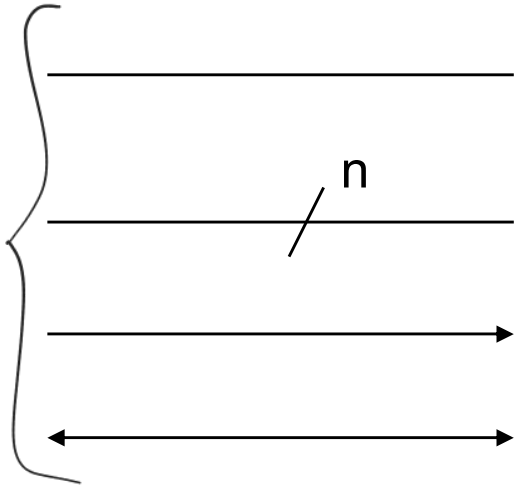
Lezione 15: BUS

Componenti esterni al core del processore

- **BUS = veicolo (lento) nel quale molte persone viaggiano … ed inoltre…**
- **Un insieme di fili…**



= INSIEME DI FILI



filo

n fili

possono anche essere
Fili direzionali

filo che trasmette in una direzione

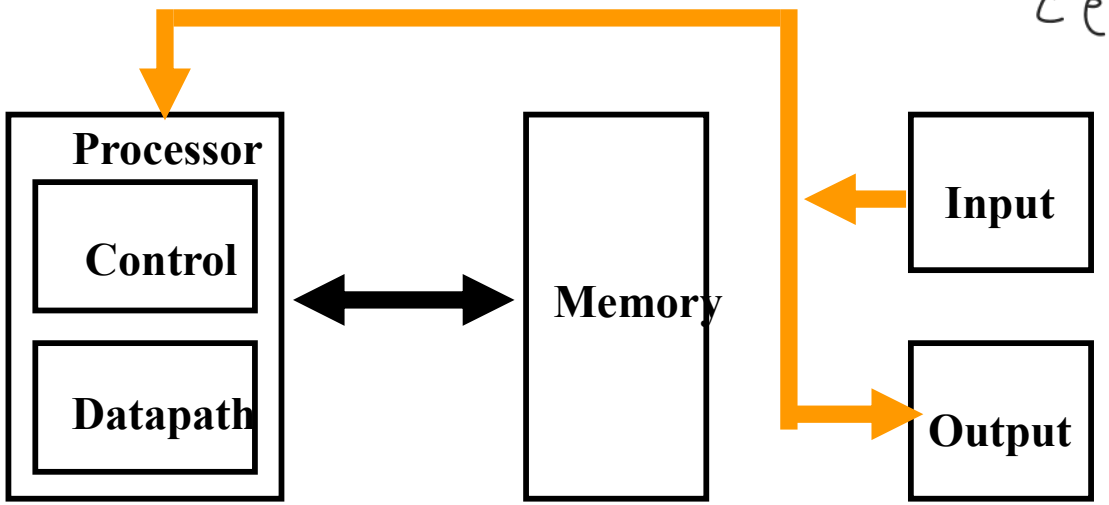
filo che trasmette in entrambe le direzioni

condiviso da più utenti

es. Porta seriale
HALF DUPLEX

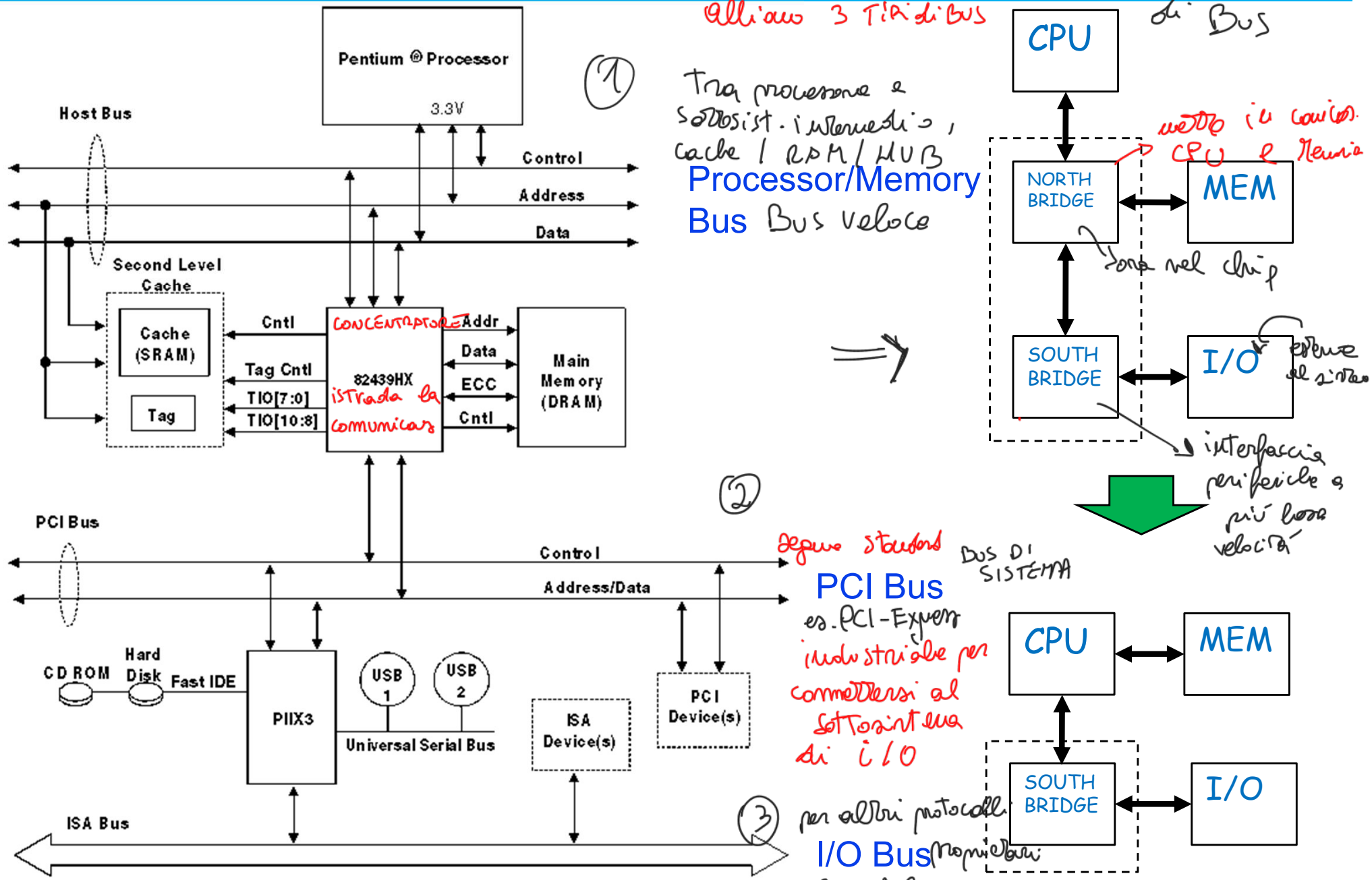
Un bus è:

- Un collegamento condiviso per comunicare
- Un insieme di fili usato per connettere vari sottosistemi

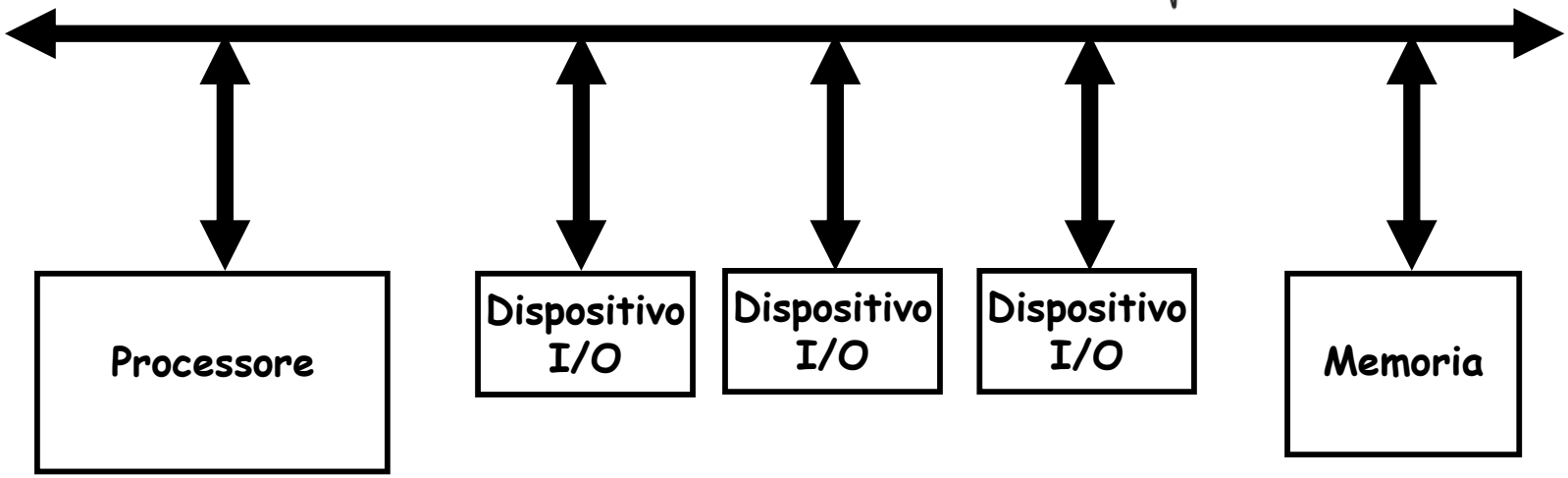


- Un bus è anche il mezzo fondamentale per costruire sistemi anche grandi e complessi
  - E' uno strumento di astrazione molto utilizzato

Esempio: Organizzazione di Sistema con CPU Pentium



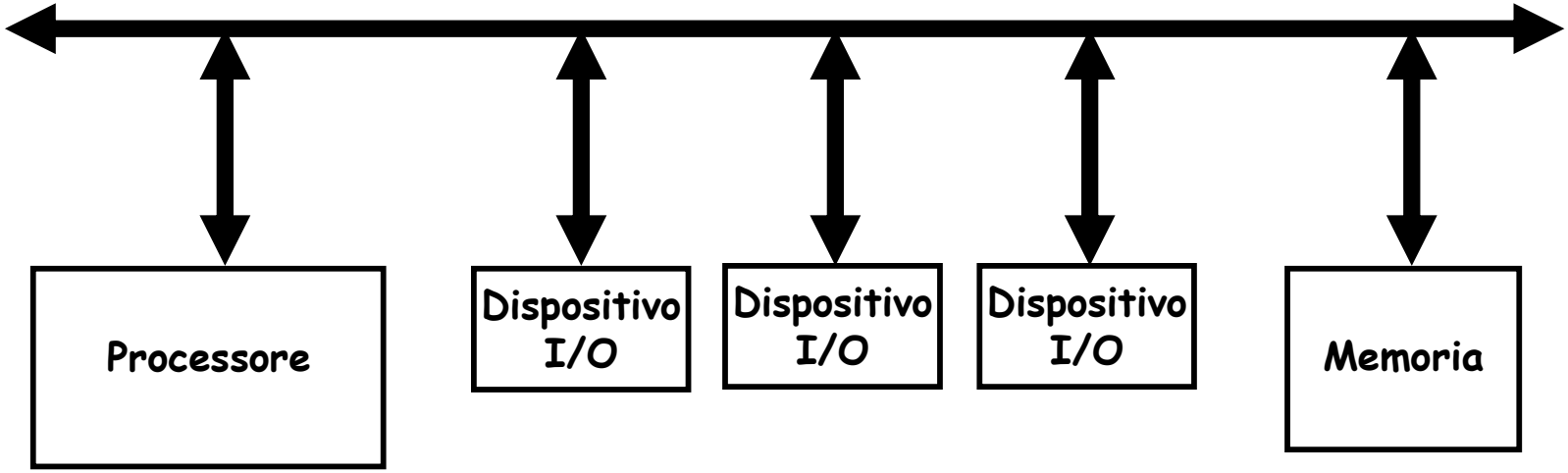
Vantaggi dei bus



- è semplice
- **Versatilità**
  - Si possono aggiungere facilmente nuovi dispositivi
  - Una data periferica può essere usata su un calcolatore diverso che usa un bus che segue lo stesso standard
- **Basso costo**
  - Un unico insieme di fili può essere condiviso da dispositivi di vario tipo
- **Gestire la complessità** suddividendo il sistema

Svantaggi dei bus

più elementi collego al bus, diminuisce la banda, ovvio un collo di bottiglia



- più elementi collegati al bus minore sarà la banda disponibile
- **Genera un "collo di bottiglia" nelle comunicazioni** ognuno avrà poco, posso saturarlo se troppi dispositivi, finisce la banda, si satura
  - **La banda del bus limita il massimo throughput ottenibile per l'I/O**
- **La velocità massima del bus è fortemente limitata da** essendo semplice
  - **Lunghezza** del bus troppo veloce => lontano Struttura semplice. Se voglio Bus veloce deve essere corto
  - **Numero** di dispositivi collegati al bus
  - **Necessità** di supportare un ampio numero di dispositivi aventi semplice non può supportare c/o velocità
    - Tempi di latenza ampiamente variabili di varie velocità periferiche
    - Velocità di trasferimento dati ampiamente variabili es. tra Tastiera e DRIVE, V dovrebbe adattarsi in base alle velocità dei vari dispositivi.

NON POSSO ADATTARMI alle VARIE VELOCITÀ dei VARI DISPOSITIVI, allora faccio più BUS

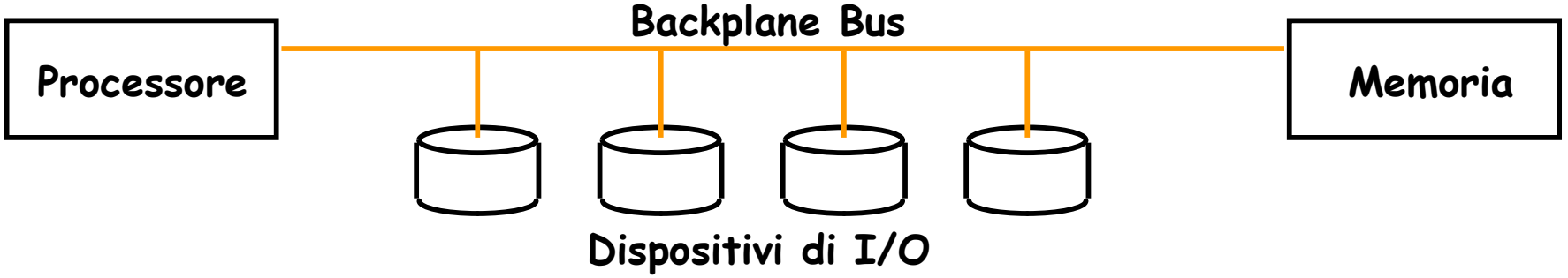
Tipi di Bus

- **Bus Processore-Memoria** (specifico del sistema)
  - Corto e ad alta velocità
  - Deve solo essere adatto al sistema di memoria utilizzato
    - Tipicamente cerca di massimizzare la banda processore-memoria
  - Connette direttamente il processore
  - Ottimizzato per i trasferimenti di blocchi di cache
- **Bus di I/O Bus** (standard industriale)
  - Tipicamente abbastanza lungo e lento
  - Deve essere adatto a collegare dispositivi di I/O molto diversi fra loro
  - Si connette al bus processore-memoria o al backplane bus
- **Backplane Bus** (standard o proprietario)
  - Backplane: interconnessione che si svolge all'interno dello chassis
  - Consente al processore, alla memoria e ai dispositivi di I/O di coesistere e cooperare fra loro
  - Si può ottenere un risparmio se si usa un solo bus per tutti i componenti del sistema

CONSENTE DI CONNETTERE TUTTI I DISPOSITIVI SULLA MOBO

Un Calcolatore con un solo Bus: Backplane Bus

Bus che permette di condividere tutti i dispositivi sulla scheda madre



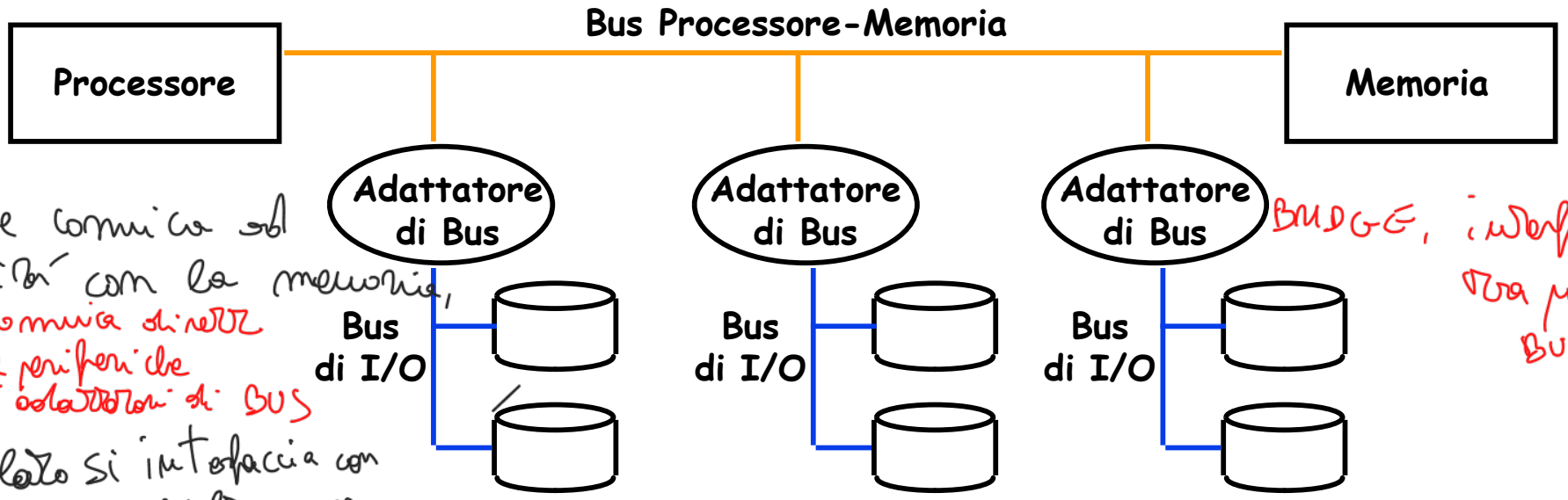
- E' basato su un singolo bus di sistema (backplane-bus) → portabile, un unico BUS
  - Gestisce la comunicazione processore/memoria
  - Gestisce la comunicazione processore/dispositivi
  - Gestisce la comunicazione dispositivi/memoria

Uso per sistemi embedded, non prevedono grandi possibilità di espansione

- Vantaggi: semplicità e a basso costo — usato nei sist embedded, a bassa velocità
- Svantaggi:
  - i) può facilmente andare in saturazione per troppo traffico diventando così un collo di bottiglia per l'intero sistema — Saturazione e bassa velocità
  - ii) dovendo rispettare la velocità di ogni componente può essere lento
- Esempi: i primi PC della IBM usavano questo tipo di architettura

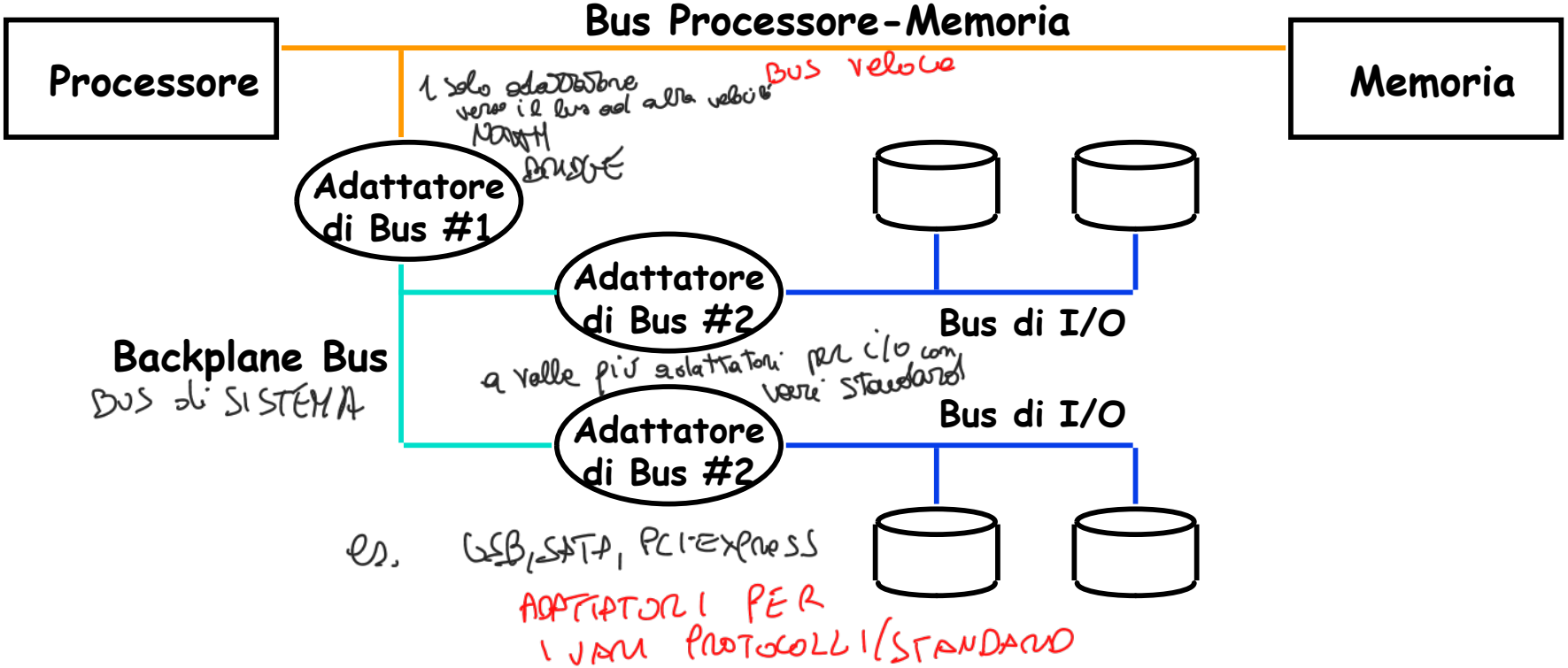
BUS ISA

Sistema con due bus



- In questa configurazione abbiamo due tipi di bus dedicati
  - La comunicazione fra i due tipi di bus avviene con dei componenti detti «adattatori di bus» o «bridge»
  - Il bus processore memoria gestisce traffico ad alta velocità
  - I bus di I/O gestiscono traffico specifico a seconda della tipologia di dispositivo e permettono l'espansione del sistema (es. PCI, SATA)
- Esempio commerciale/storico: il Macintosh-II della Apple
  - Come bus processore-memoria venne introdotto lo standard «NuBus»
  - Come bus generico di I/O venne utilizzato lo standard SCSI

Sistema con tre bus



- In questo caso il bus processore-memoria parla con un unico adattatore verso il backplane-bus restando così meno carico e più veloce
- Il backplane-bus serve per far parlare fra lor vari tipo di adattatori
- I bus di I/O sono disaccoppiate e dedicati come nel caso precedente (es. PCI, SATA, …)

Come si definisce un bus? Cosa caratterizza un BUS?

Freccia riferimento ad uno stack complesso, una serie di strati con funzionalità, che comunica con i precedenti

Più astratto, es. se voglio scrivere, ci sarà una transazione di scrittura tramite vari segnali guidati da un clock

ogni transazione implicherà l'attivazione di segnali per determinati intervalli di Tempo, agganciati ad un clock

Protocollo delle Transazioni

es. se voglio scrivere o leggere implicherà l'accensione di alcuni segnali, scatto di segnali elettrici

Specifica Temporizzazioni e Segnali

Diagramma Temporale

dobbiamo specificare le Temporizzaz. e verificare un diagramma Temporale relativo a ciò che viene trasferito su questi fili

Insieme di fili

I definiamo i fili di comunicaz.

Scatto segnali elettrici su dei fili

Specifiche Elettriche

es. 0 logico
1 tra +15,5 volt.

Caratteristiche fisiche e meccaniche, e tipi di connettori

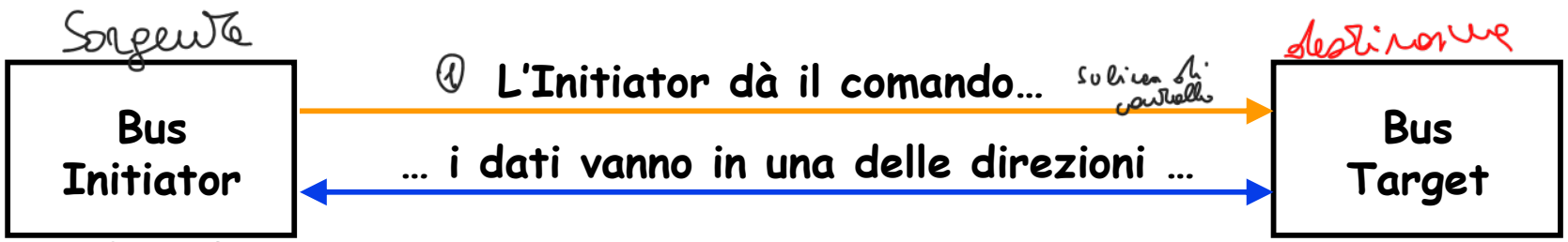
connettore a Spina ecc.
es. RS 232, tipo connettore

Prima distinzione dei fili del bus:

Tipi di fili: le linee di controllo e le linee dati

- **Linee di Controllo**
  - Servono sia per indicare quali tipi di dato sono trasferiti…
  - …che per indicare il tipo di segnalazione (es. request/acknowledgment nel caso asincrono o clock nel caso sincrono)
  - … oppure la direzione (es. lettura oppure scrittura)
  - Tipicamente sono un numero basso di fili (es. 2-10)
- **Linee Dati**
  - Servono per trasportare le informazioni vere e proprie
    - Es. l'indirizzo a cui fare una scrittura/lettura
  - L'informazione può essere anche un valore (es. a 64 o 128 bit) da scrivere
  - L'informazione può essere anche l'identificatore della sorgente-dati (source) o del destinatario-dati (destination)
  - L'informazione può essere anche un comando complesso

Initiator/Target e Transazioni



- Una transazione sul bus include tre fasi
  - Decidere chi è l'Initiator – fase di arbitraggio
  - Dare il comando (o l'indirizzo) – fase di richiesta
  - Trasferire i dati – fase di azione

  } Come funziona una transazione

- L'Initiator è l'unità che:
  - Ha il controllo del bus
  - Inizia la transazione dando un comando o specificando un'indirizzo
- Il Target è l'unità che:
  - Viene attivata dalla transazione ovvero risponde alla richiesta
  - Invia i dati all'Initiator, se l'Initiator richiede dati
  - Preleva i dati inviati dall'Initiator, se l'Initiator vuole mandare dati

Altra distinzione di Bus

Bus Sincroni e Bus Asincroni

- **Bus Sincrono**:
  - Fra le linee di controllo c'è il segnale di clock
  - Il protocollo di comunicazione è fisso ed è riferito al clock
  - Vantaggio: richiede pochissima logica e va molto veloce
  - Svantaggi:
    - Ogni dispositivo sul bus deve andare alla stessa frequenza di clock
    - Per evitare il "clock skew" per avere bus lunghi devo abbassare le velocità

richiede poca logica ed è veloce

Ci si aggancia al clock

Impongo che tutti i dispositivi obbediscano alla stessa freq di clock

IMPONGO che tutti i dispositivi che vengono collegati obbediscano a quel tipo di clock, stessa velocità

↳ per fili di diversa lunghezza, avrò dei ritardi

- **Bus Asincrono**:
  - Sono sempre necessarie due linee di controllo (req, ack) per gestire il trasferimento (protocollo di handshaking)
  - Non ha bisogno del clock
  - Può collegarsi alla quasi totalità dei dispositivi
  - Può essere anche abbastanza lungo senza problemi di "clock skew"

↳ Fasi diverse

posso accordarmi qualsiasi velocità del dispositivo

↳ MOLTO FLESSIBILE, MA PIÙ LENTO

- Non riesco ad andare veloce come il clock

PROTOCOLLO DI HANDSHAKING INTRODUCE RITARDI

se ho più fili, più lunghi, più conti, un segnale arriva prima ecc introduce problemi di Temporizzazioni

3 Fasi: Arbitraggio → comando → azione

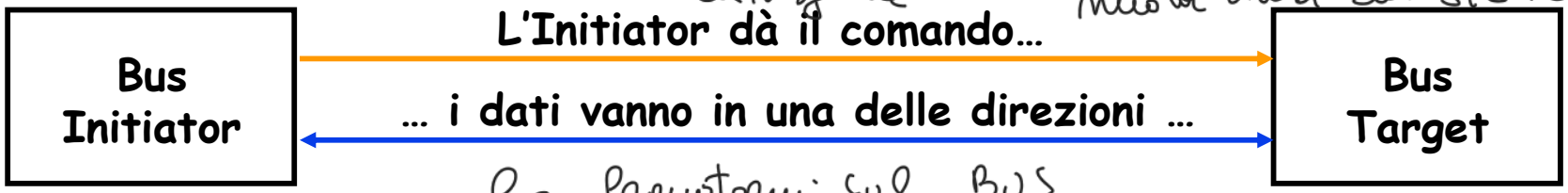
Arbitraggio: ottenere accesso al Bus

Metodo per fare accesso al bus.

Metodo per decidere chi è l'INITIATOR

Devo decidere chi è la seguente

Come prenotarsi? nuova unità del sistema



Per prenotarmi sul BUS

Faccio richiesta all'arbitro

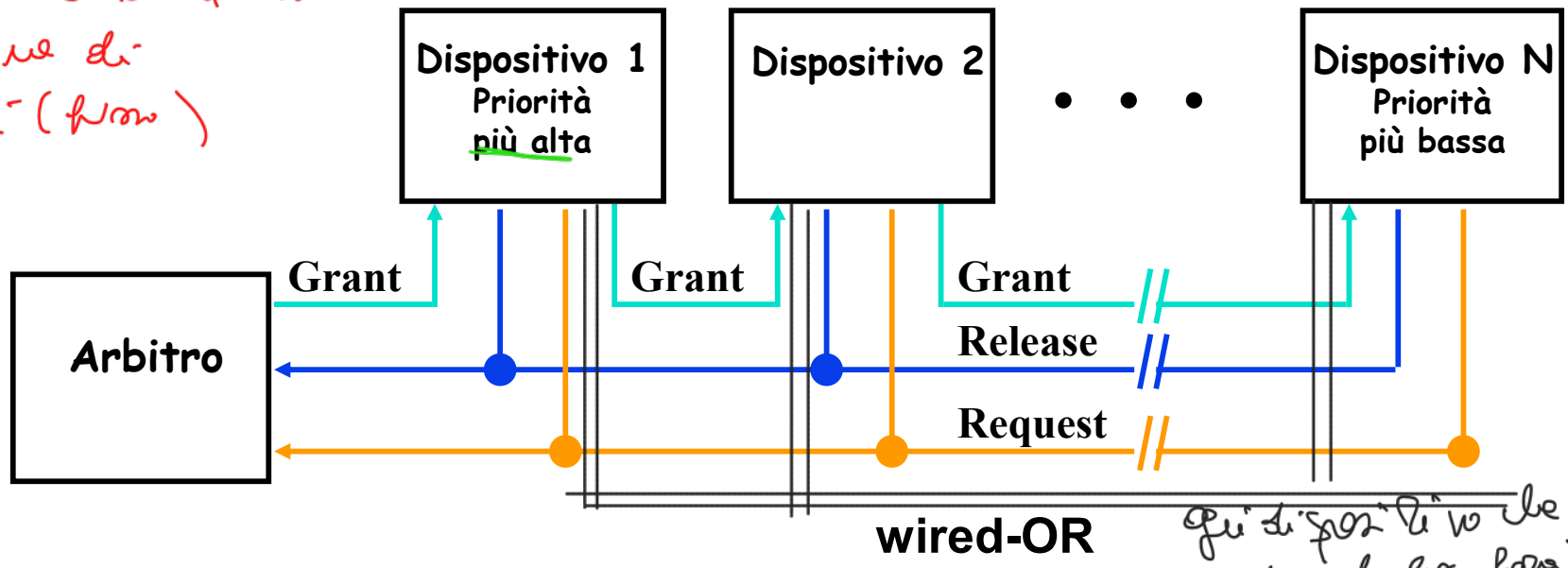
- E' uno dei problemi più importanti nella progettazione di un bus
  - Come viene prenotato il bus da un dispositivo che desidera usarlo?
- Innanzitutto per semplificare la situazione si deve arrivare alla definizione di chi è l'Initiator e chi è il Target
  - A quel punto solo il bus-Initiator potrà controllore l'accesso al bus
    - Tutte le successive richieste saranno iniziate e controllate da lui
  - Un Target risponderà alle richieste leggendo o scrivendo qualcosa OBBEDENDO al COMANDO
- Il sistema più semplice, che usa questo schema, potrebbe funzionare così: es. No arbitro, lo fa il processore
  - Il processore è il solo bus-Initiator
  - Tutte le richieste saranno controllate dal processore
  - Principale svantaggio: il processore è coinvolto in ogni transazione invece lo interrompo spesso

fare cercare sleve occupati dell'arbitraggio

Bus-Initiator multipli: necessità di Arbitraggio

- Schema di arbitraggio del bus
  - Un bus-Initiator che voglia usare il bus segnala la richiesta ad un Arbitro
  - Il bus-Initiator non può usare subito il bus resta in attesa del permesso (grant)
  - Il bus-Initiator dovrà infine segnalare all'Arbitro che ha finito di usare il bus
- L'Arbitro cerca di bilanciare due fattori
  - Priorità della richiesta: il dispositivo a priorità maggiore sarà servito per primo
  - Fairness: anche se un dispositivo ha bassa priorità bisogna garantire che questo possa ottenere il controllo del bus entro un tempo ragionevole
- Categorie principali degli schemi di arbitraggio
  - Arbitraggio Centralizzato Daisy-Chain
  - Arbitraggio Centralizzato Parallelo
  - Arbitraggio Distribuito con Auto-Selezione
  - Arbitraggio Distribuito con Rilevamento delle Collisioni

Arbitraggio Centralizzato Daisy Chain

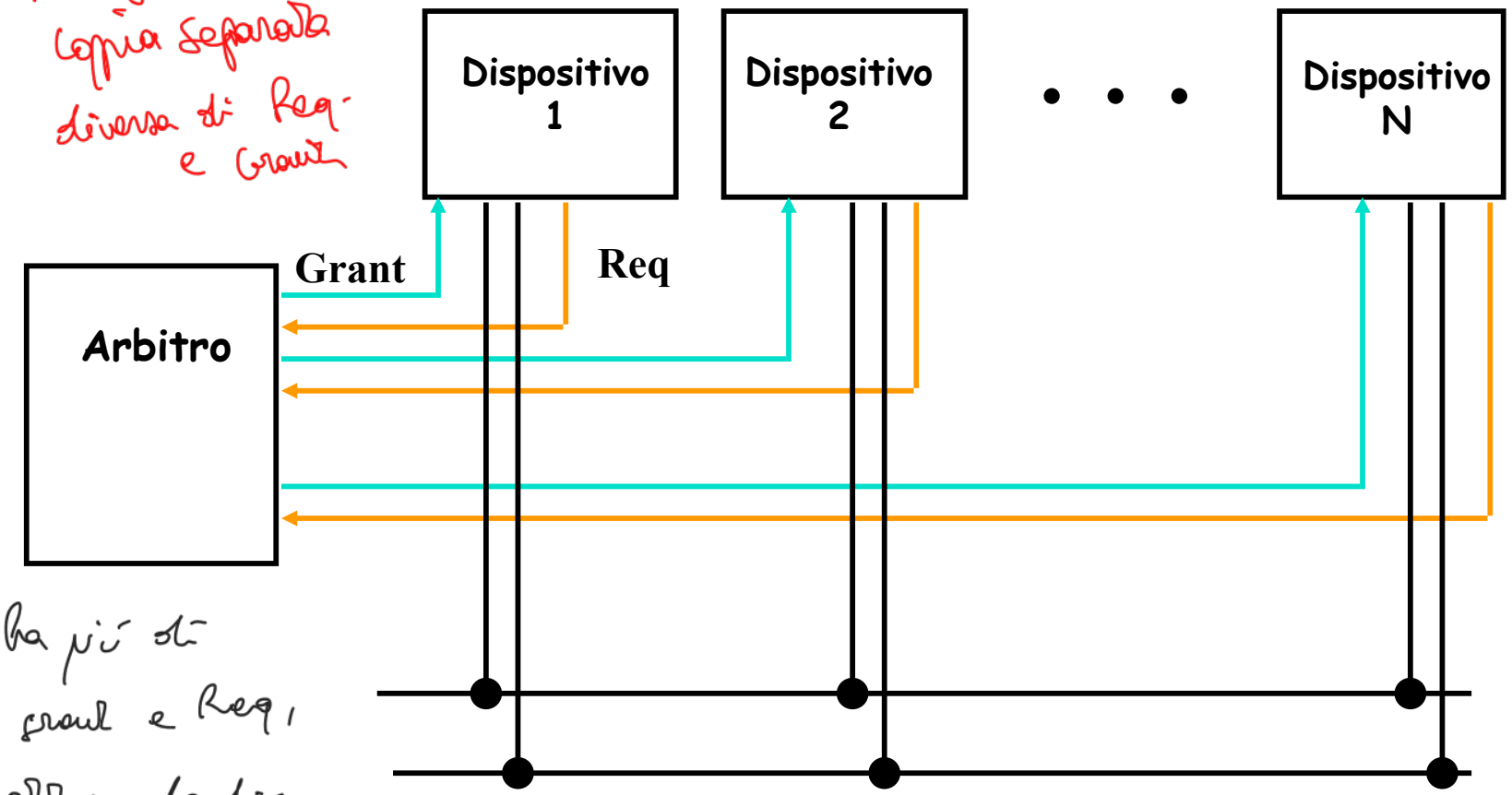


- Vantaggio: semplice
- Svantaggi:
  - Non è possibile garantire la fairness:
    - Un dispositivo a valle potrebbe rimanere bloccato indefinitamente
  - Il tempo di propagazione del segnale di grant può limitare la velocità del bus
- Es.: VMEbus

Arbitraggio Centralizzato Parallelo

ogni dispositivo ha una copia di fili di Req, grant (e Release (?))

per ogni dispositivo copia separata diversa di Req e Grant



Arbitro decide chi ha chiesto il grant

Se arbitro ha più di 1 porta grant e Req, posso connettere 1 disp.

- Usato nella quasi totalità dei bus processore-memoria e nei bus di I/O ad alta velocità
- Inoltre è usato nei bus: Multibus I, EISA, PCI

gli slot PCI sono quelli non posso aggiungere altri

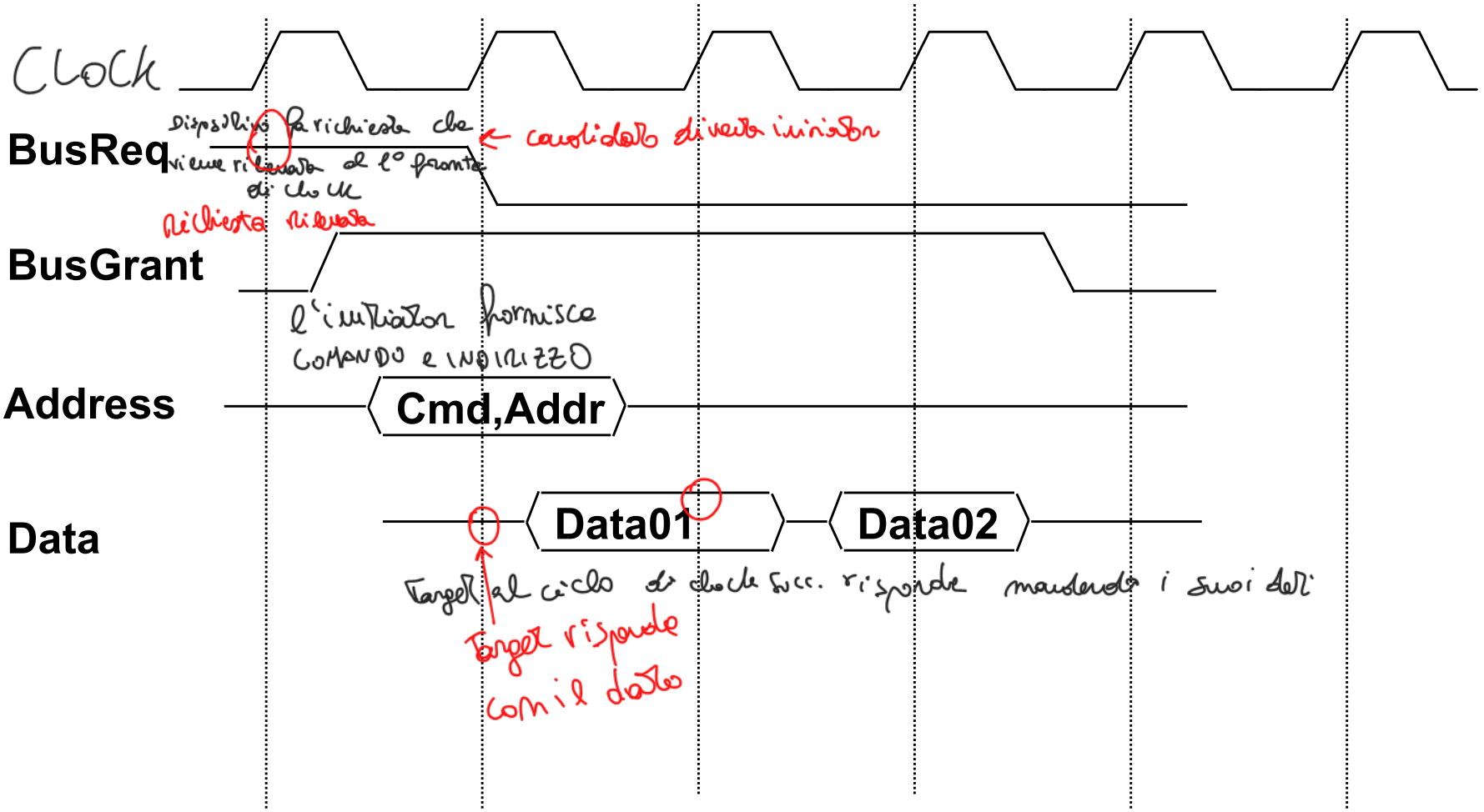
non è più facilmente scalabile se inserisco più dispositivi
Se arbitro ha 1 porta qui dispositivi ci attacco

Arbitraggio Distribuito

è dinamico

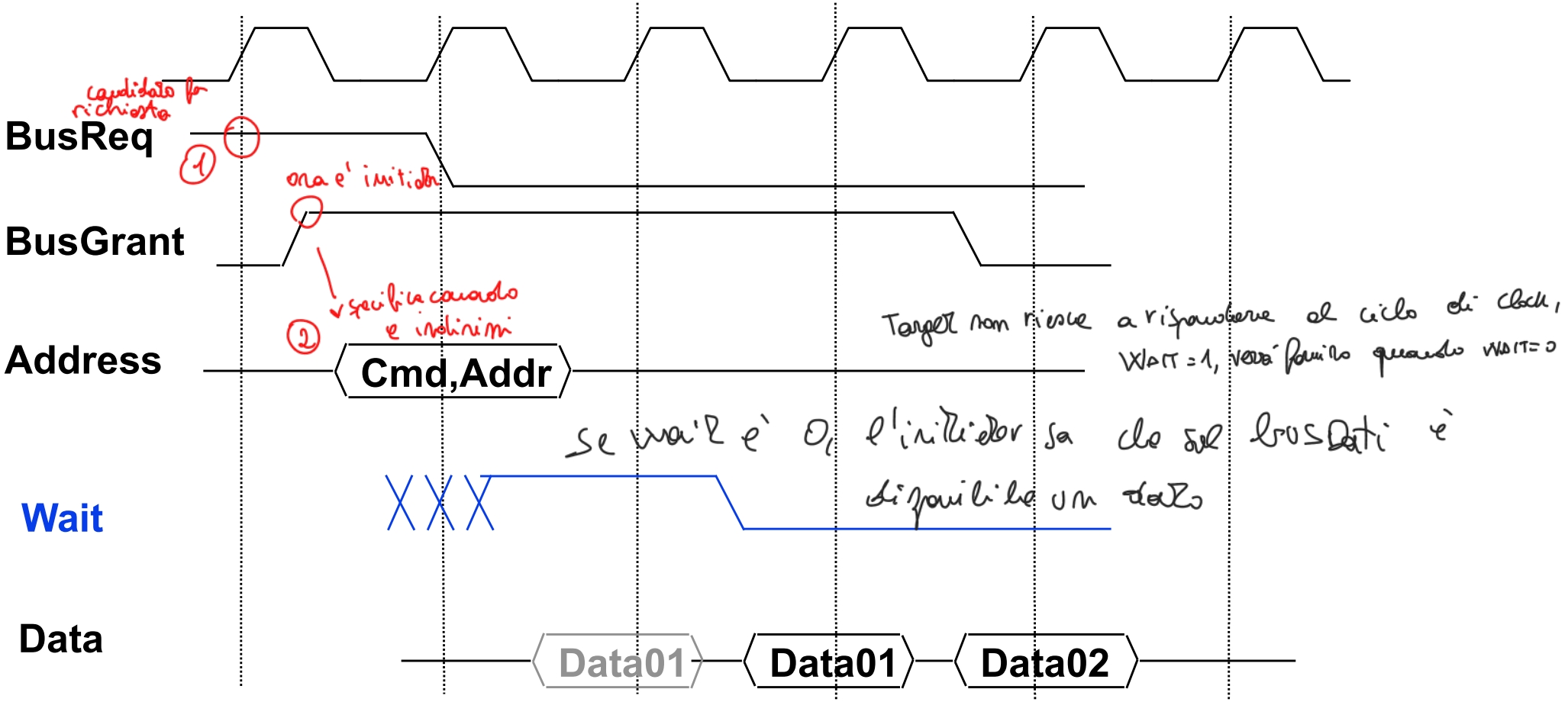
- La scelta dell'Initiator non viene fatta da una entità precisamente identificabile
- Esempi:
  - Multibus I: arbitraggio distribuito in daisy-chain
    - Inizialmente il bus è controllato da un'unità a priorità intermedia M
    - La richiesta da parte di un'unità A a priorità più alta attiva la logica per "convincere" M al rilascio del bus
  - Ethernet: arbitraggio distribuito con rilevamento delle collisioni
    - Inizialmente un'unità detiene il controllo del bus ma lo farà solo per un tempo limitato
    - Un potenziale Initiator prova a trasmettere: se il bus è libero bene, altrimenti riprova più tardi
    - Il tempo di attesa si calcola con una legge di decadimento casuale esponenziale, per evitare che si ripeta la stessa combinazione di unità che tentano l'accesso

Protocollo Sincrono (Caso Semplice)



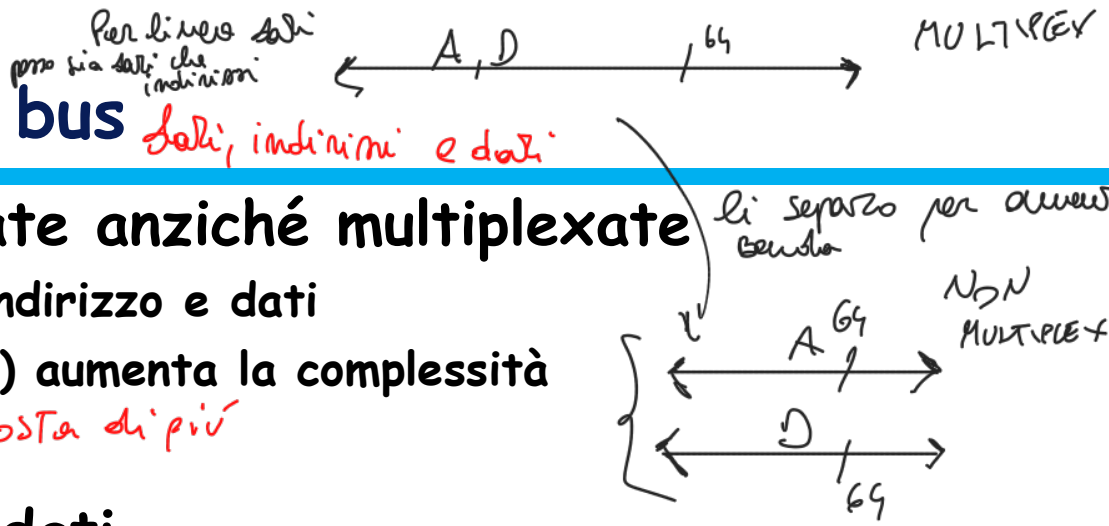
- Anche un bus di memoria è più complesso di questo
  - La memoria (Target) può prendere più cicli per rispondere
  - Dovrà quindi segnalare che non è in grado di accettare indirizzi…

Protocollo Sincrono (Caso Tipico)



- Esempio: l'Initiator specifica un indirizzo (Address) da cui vuole leggere
- Con WAIT=1 il Target segnala che NON è pronto al trasferimento dati e che quindi i dati (es. Data01) verranno forniti successivamente
- Poi (con WAIT=0) il trasferimento ha luogo alla velocità del bus

Come aumentare la banda del bus

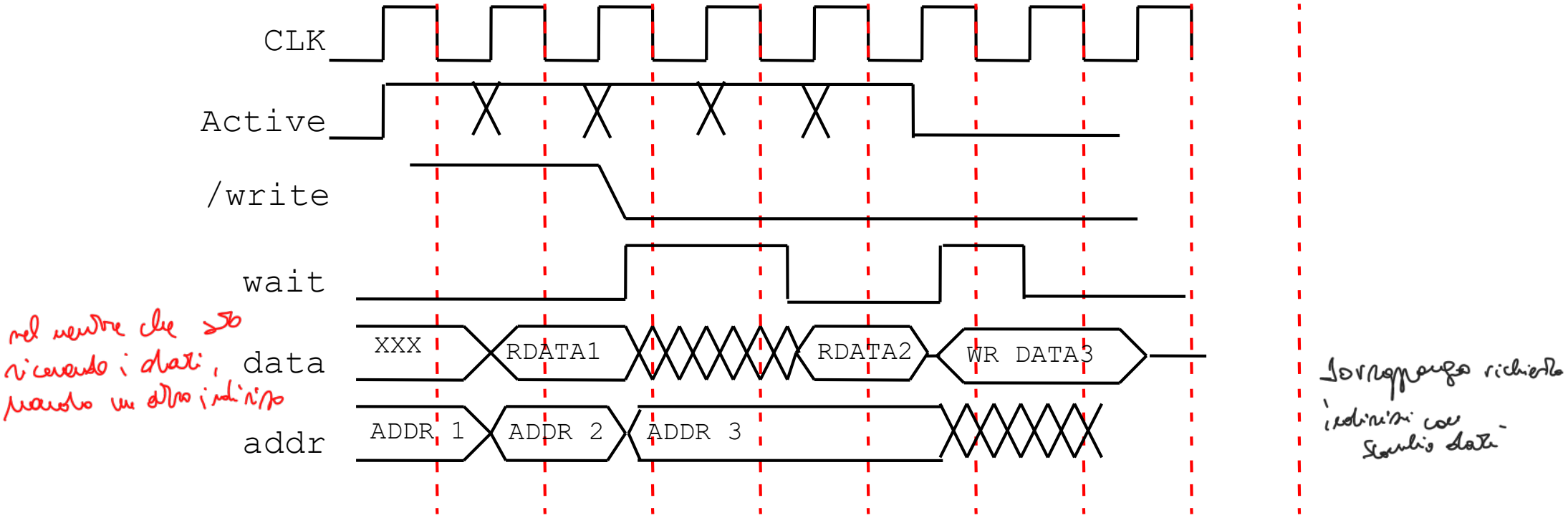


- **Uso di linee indirizzi/dati separate anziché multiplexate**
  - **posso trasmettere nello stesso ciclo indirizzo e dati**
  - **Svantaggi: (a) necessito di più fili, (b) aumenta la complessità**
- **Aumentare la larghezza del bus dati**
  - **Il trasferimento di più word richiederà meno cicli di bus**
  - **Esempio: nella SPARCstation 20 il bus di memoria è a 128 bit**
  - **Svantaggio: ancora necessito di più fili**
- **Trasferimento a blocchi (burst transfer)**
  - **Evito di ripetere l'indirizzo quando trasferisco K word successive**
  - **Specifico solo l'indirizzo della prima word**
  - **Il bus non viene rilasciato finché l'ultima word non è arrivata**
  - **Svantaggi: (a) maggiore complessità, (b) aumento del tempo di risposta nei casi di singola richiesta**

Protocolli basati sul concetto di "Pipeline"

- **Durante la fase di accesso ai dati inizio già a specificare l'indirizzo del trasferimento successivo**

Esempio con singolo Initiator (processore-cache)



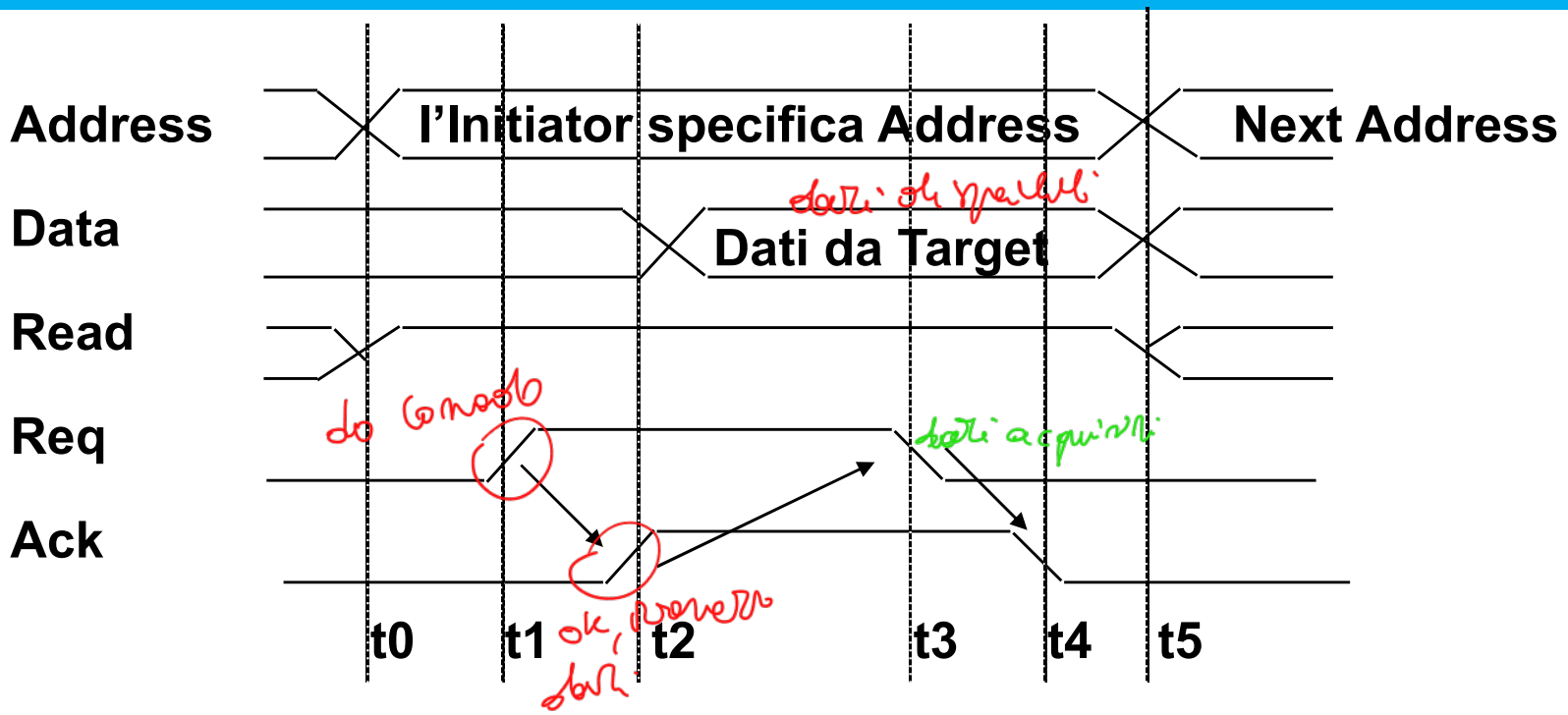
Aumento delle Transazioni su Bus MultiInitiator

- **Arbitraggio sovrapposto** (overlapped)
  - Si guadagna tempo effettuando la fase di arbitraggio per la transazione successiva sovrapposta a quella in corso
- "**Bus parking**"
  - L'Initiator mantiene il bus ed effettua varie transazioni senza passare nuovamente alla fase di arbitraggio, fino a che qualche altro Initiator non si fa avanti
- **Sovrapposizione delle fasi di indirizzamento e accesso**
  - Visto nella pagina precedente
- "**Split-phase**" (o "packet switched") bus
  - Fasi di indirizzamento e accesso completamente separata
  - Ognuna necessita di un arbitraggio separato
  - Durante l'indirizzamento si produce anche un identificatore (tag) del pacchetto, che verrà associato ai dati in fase di risposta, per permettere l'abbinamento indirizzo-dato
- **Nei bus moderni si usano tutte le tecniche precedenti combinate fra loro**

Handshake Asincrono: Scrittura

- t0: L'Initiator ha ottenuto il controllo e specifica l'indirizzo, la direzione e dati; attende poi un certo intervallo di tempo affinché il Target capisca di essere coinvolto in quel trasferimento…
- t1: L'Initiator attiva il filo "Request"
- t2: Il Target attiva il filo "Ack", per indicare di aver ricevuto i dati → ha finito
- t3: L'Initiator rilascia "Req" → ha capito che il Target ha finito
- t4: Il Target rilascia "Ack" → ha capito che l'Initiator ha capito
- t5: la transazione è terminata e si può cambiare indirizzo/dati/r-w

Handshake Asincrono: Lettura



- t0: L'Initiator ha ottenuto il controllo e specifica l'indirizzo, la direzione e i dati; attende poi un certo intervallo di tempo affinché il Target capisca di essere coinvolto in quel trasferimento…
- t1: L'Initiator attiva il filo "Request"
- t2: Il Target attiva il filo "Ack", per dire che è pronto a trasmettere i dati
- t3: L'Initiator rilascia "Req" avendo ricevuto i dati → ha finito
- t4: Il Target rilascia "Ack" → ha capito che l'Initiator ha finito
- t5: la transazione è terminata e si può cambiare indirizzo/dati/r-w