Bus

Introduzione

Protocolli logici

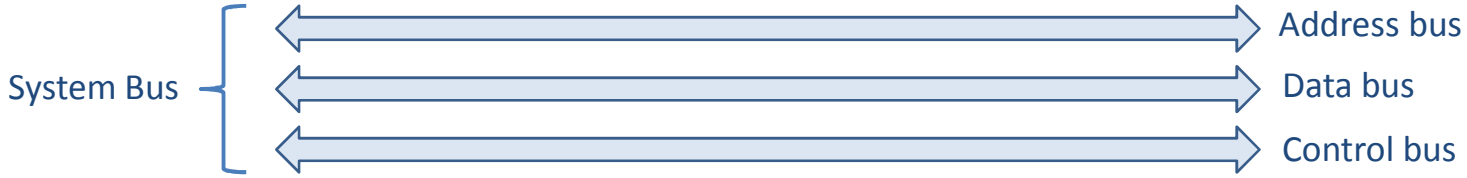
Bus asincrono

Bus sincrono

Arbitraggio

Introduzione

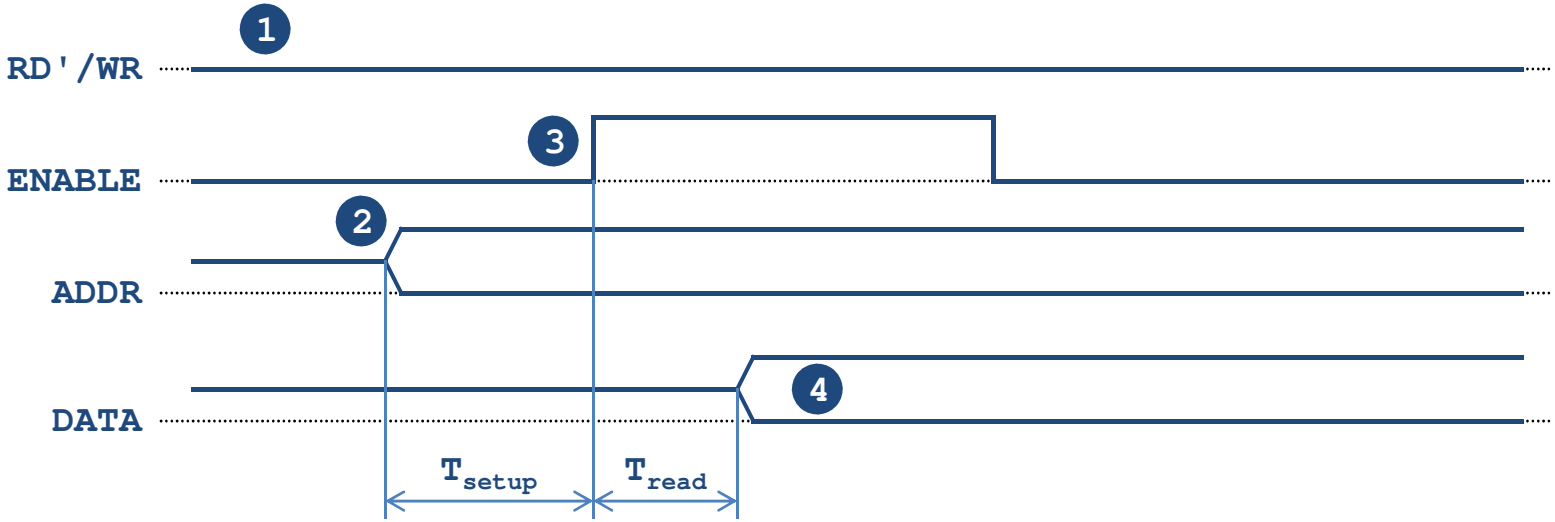
- **Il bus di sistema connette il processore**
  - Alla memoria centrale
  - Alle periferiche
- **Si tratta di un insieme di linee logicamente suddivise in**
  - Bus indirizzi o address bus, destinato a contenere unicamente indirizzi
  - Bus dati o data bus, destinato a contenere dati
  - Bus di controllo o control bus, raccoglie tutte le linee di controllo dei dispositivi



- **In questo caso parliamo di bus parallelo**
  - Ogni informazione dispone di una linea dedicata
- **Esistono tuttavia bus di sistema seriali**
  - Una (o poche) linee portano tutte le informazioni, in istanti diversi di tempo, cioè in maniera seriale

Introduzione

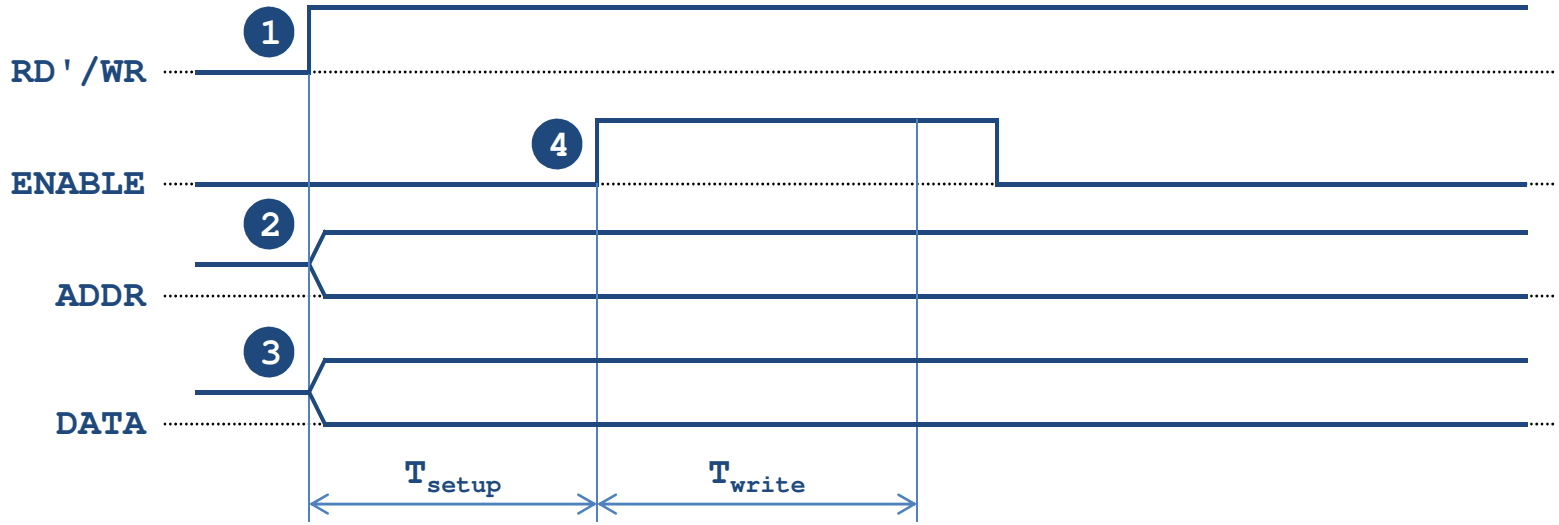
- **Schema concettuale di base del ciclo di lettura**



1. Il processore asserisce il segnale RD'
   - Inizia il ciclo di lettura
2. Il processore pone l'indirizzo sul bus indirizzi ADDR
3. Il processore asserisce il segnale ENABLE
   - Almeno un tempo T_{setup} dopo che l'indirizzo è stabile
   - La periferica o la memoria inizia l'operazione di lettura
   - I dati sono disponibili sul bus dati non prima di un tempo T_{read}
4. Il processore preleva i dati dal bus dati DATA

Introduzione

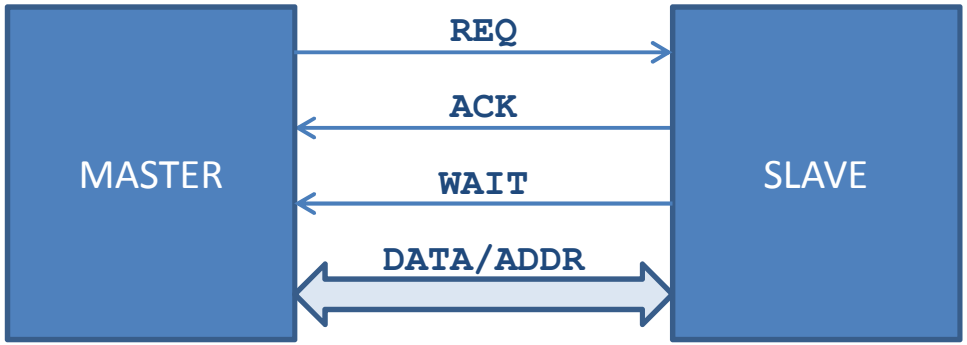
- **Schema concettuale di base del ciclo di lettura**



1. Il processore asserisce il segnale WR
   - Inizia il ciclo di scrittura
2. Il processore pone l'indirizzo sul bus indirizzi ADDR
3. Il processore pone sul bus DATA dati il dato da scrivere
4. Il processore asserisce il segnale ENABLE
   - Almeno un tempo T_{setup} dopo che l'indirizzo è stabile
   - La periferica o la memoria inizia l'operazione di scrittura
   - La scrittura sarà avvenuta non prima di un tempo T_{write}

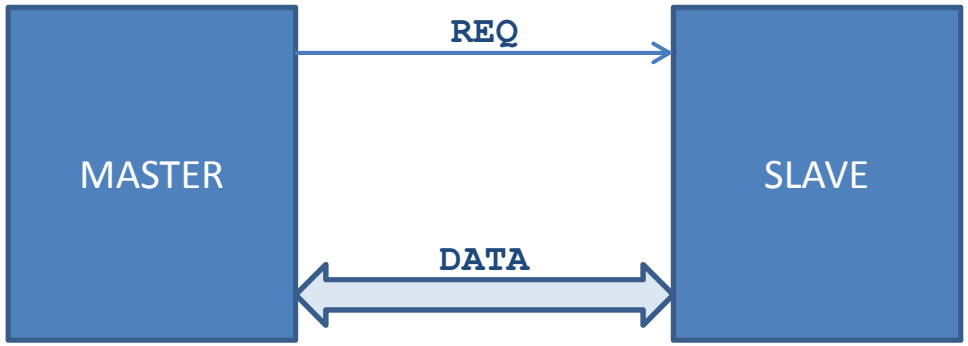
Protocolli logici

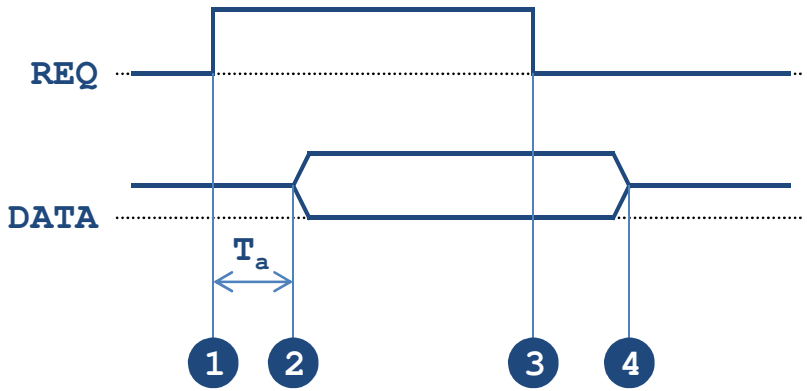
- **Consideriamo un semplice modello concettuale delle oprazioni su bus**
- **Attori**
  - Master: Il componente che gestisce la comunaicazione sul bus
  - Slave: Il componente che reagisce alle richieste del master
- **Segnali**
  - REQ: Richiesta di una operazione dal parte del master
  - ACK: Accettazione della richiesta da parte dello slave
  - WAIT: Lo slave richiede al master di attendere
  - DATA: Dati trasferiti
- **L'architettura di riferimento è pertanto la seguente**



Protocolli logici di base

- **Protocollo strobe**

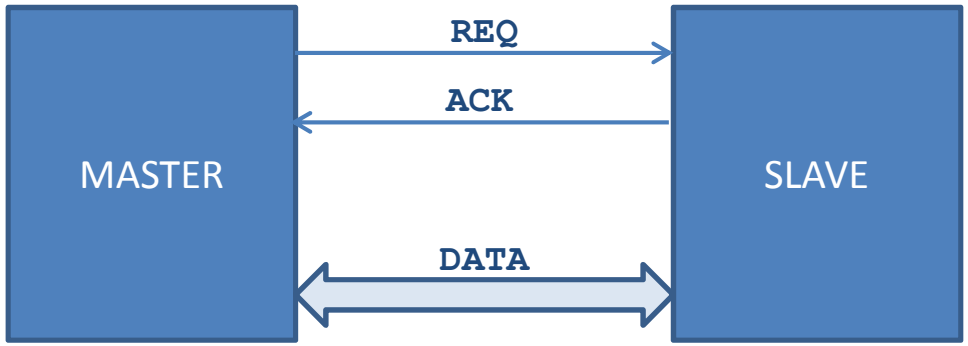


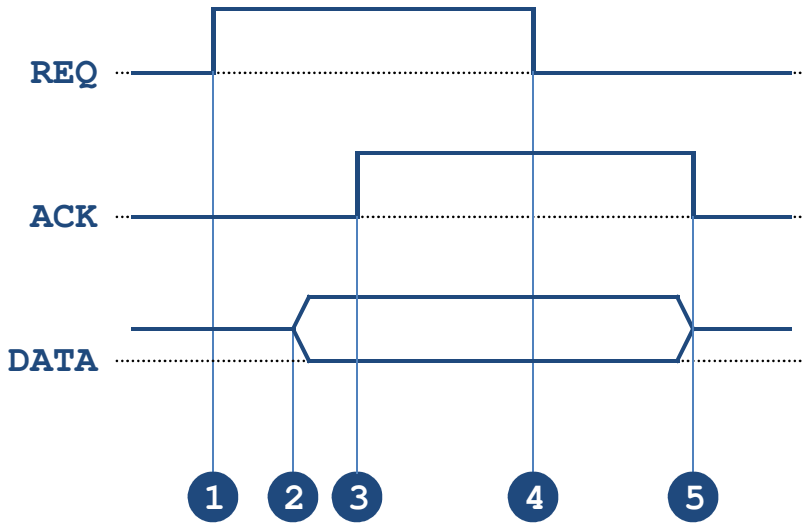


1. Il master asserisce REQ
   Inizia il ciclo di trasferimento
2. Lo slave pone i dati sul bus DATA entro T_a
3. Il master preleva i dati dal bus DATA
3. Il master deasserisce REQ
   Termina la sua richiesta
4. Lo slave è pronto per una nuova operazione

Protocolli logici di base

- **Protocollo handshake**

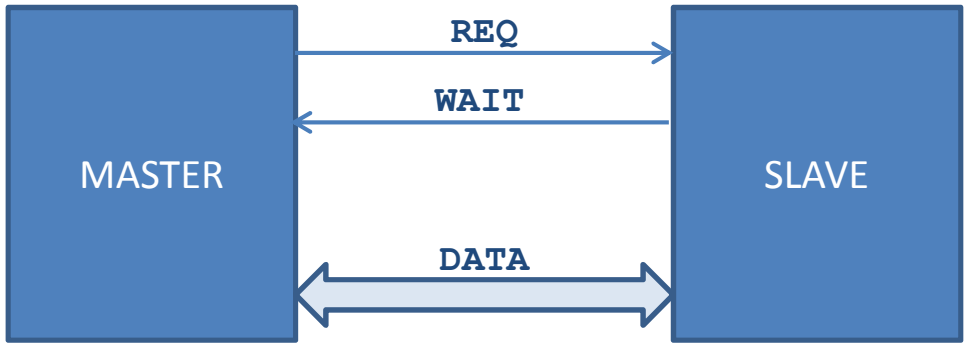


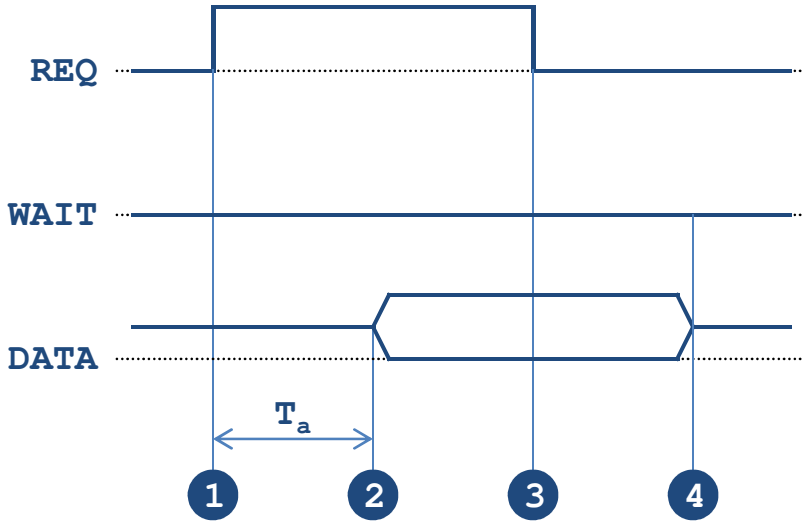


1. Il master asserisce REQ
   Inizia il ciclo di trasferimento
2. Lo slave pone i dati sul bus DATA
3. Lo slave asserisce ACK
   Segnala al master la disponibilità dei dati
3 4. Il master preleva i dati
4. Il master deasserisce REQ
   Segnala allo slave di avere ricevuto ACK
5. Lo slave deasserisce ACK
   Lo slave è pronto per una nuova operazione

Protocolli logici di base

- **Protocollo strobe/handshake, nel caso di trasferimenti veloci**

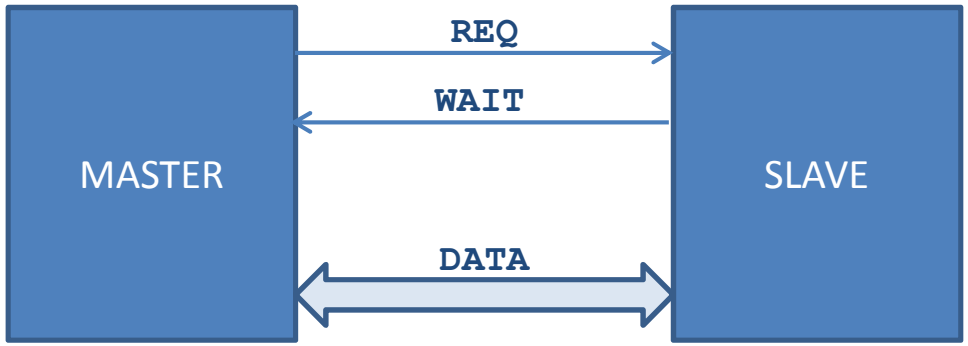


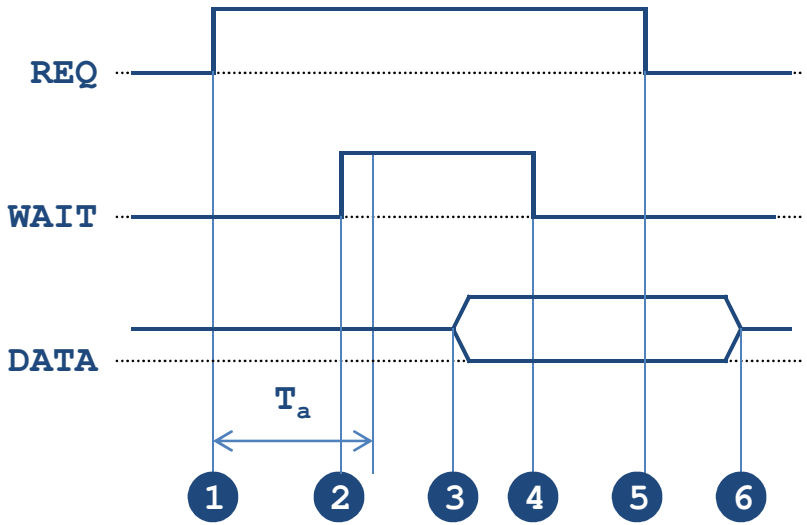


1. Il master asserisce REQ
   - Inizia il ciclo di trasferimento
2. Lo slave pone i dati sul bus DATA entro T_a
   - Il dispositivo è veloce
   - La linea WAIT non è usata
3. Il master preleva i dati dal bus DATA
4. Il master deasserisce REQ
   - Termina la sua richiesta
5. Lo slave è pronto per una nuova operazione

Protocolli logici di base

- **Protocollo strobe/handshake, nel caso di trasferimenti lenti**





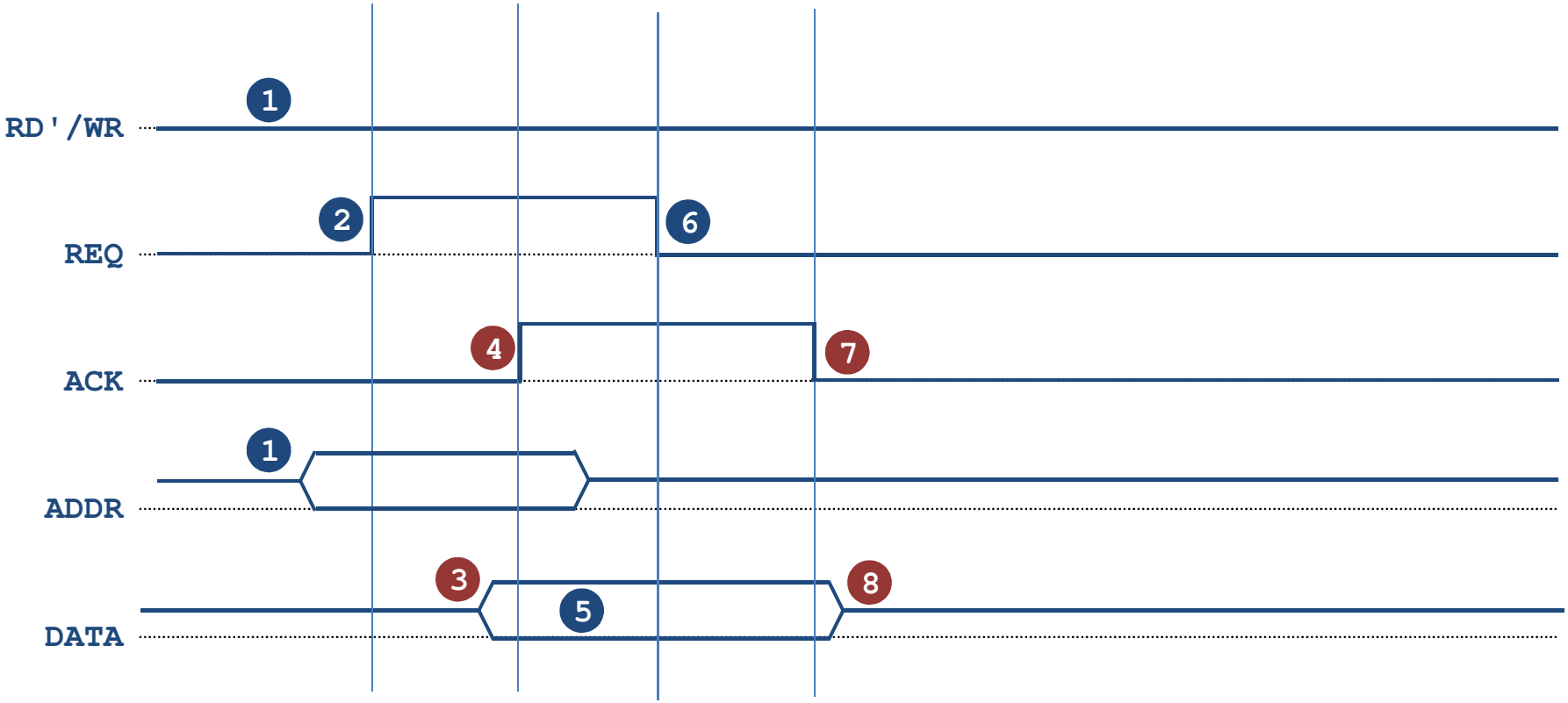
1. Il master asserisce REQ
   - Inizia il ciclo di trasferimento
2. Lo slave richiede un tempo maggiore di T_a
   - Lo slave asserisce il segnale di WAIT
   - Ciò avviene poco prima di T_a
3. Lo slave pone i dati sul bus DATA
4. Lo salve deasserisce il segnale WAIT
5. Il master preleva i dati dal bus DATA
5. Il master deasserisce REQ
   - Termina la sua richiesta
6. Lo slave è pronto per una nuova operazione

Bus sincroni e asincroni

- **La realizzazione dei protocolli logici può avvenire mediante due schemi**
  - Asincrono
  - Sincrono
- **Nello schema asincrono**
  - I segnali possono avere transizioni in qualsiasi istante
  - Questo schema è molto simile a quanto visto finora a proposito dei protocolli logici
  - Normalmente questo schema implementa il protocollo di handshake
- **Nello schema sincrono**
  - I segnali sono vincolati a cambiare solo in corrispondenza dei fronti di clock
    - Ricordiamo che il clock del bus è in genere diverso da quello del master e dello slave
  - Normalmente questo schema implementa il protocollo strobe/handshake
    - In questo caso il tempo di attesa Ta coincide con un numero prefissato di cicli di clock del bus
    - In genere tale tempo è pari ad un ciclo di clock
  - Spesso la sincronizzazione prevede che
    - Il master sia sincronizzato al fronte di salita del clock
    - Lo slave sia sincronizzato al fronte di discesa del clock

Bus asincrono

- Ciclo di lettura

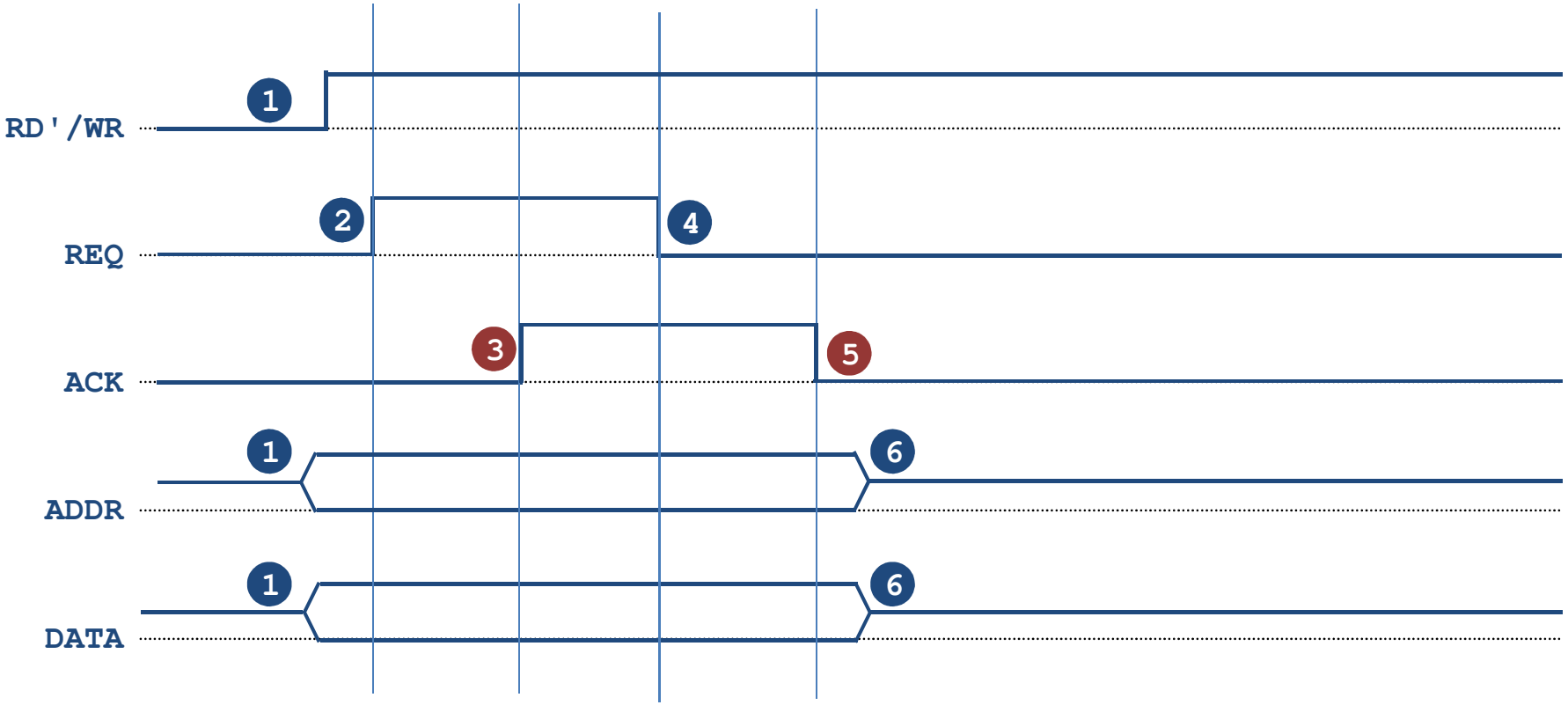


n Operazioni svolte dal bus master

k Operazioni svolte dal bus slave

Bus asincrono

- Ciclo di scrittura

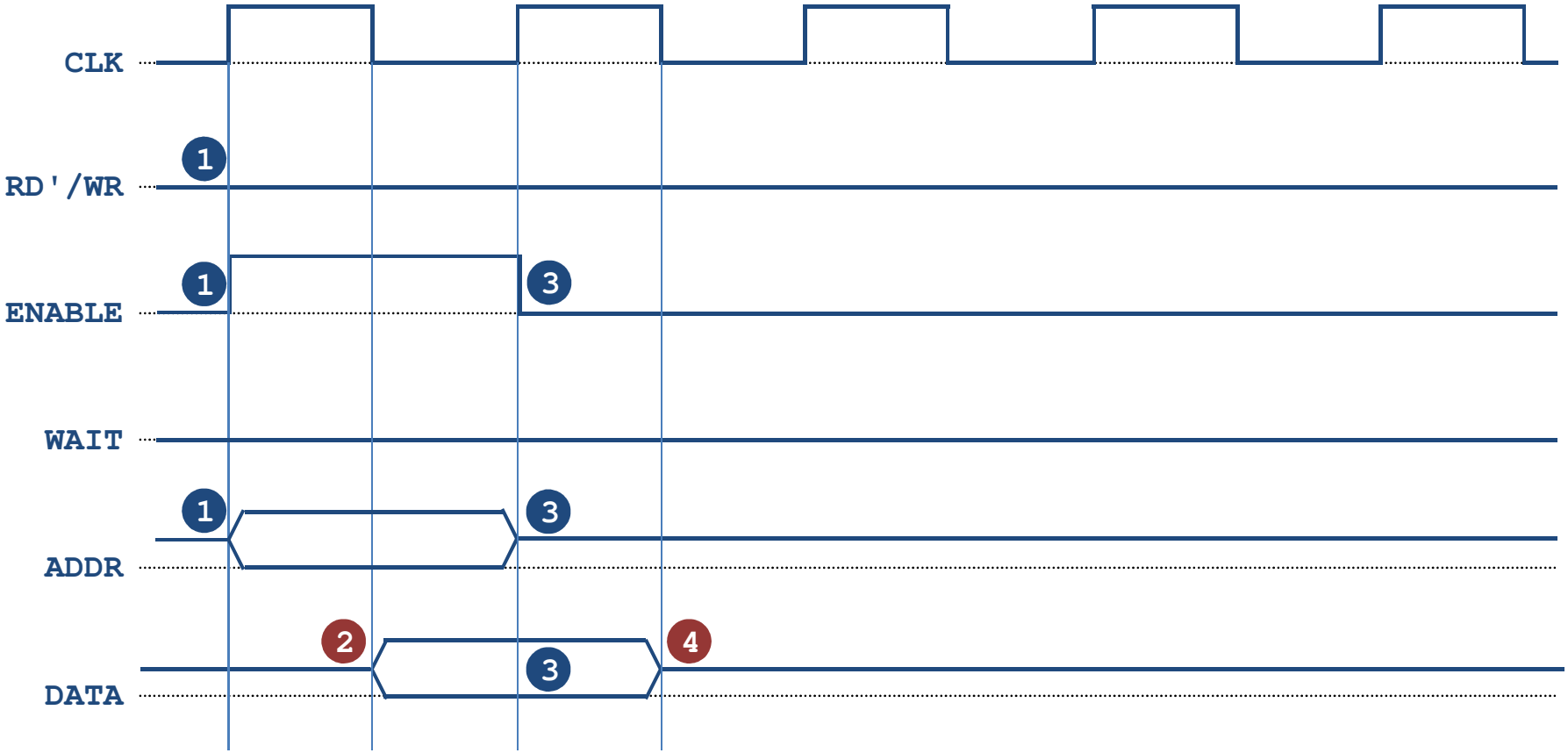


n Operazioni svolte dal bus master

k Operazioni svolte dal bus slave

Bus sincrono

- Ciclo di lettura senza wait state

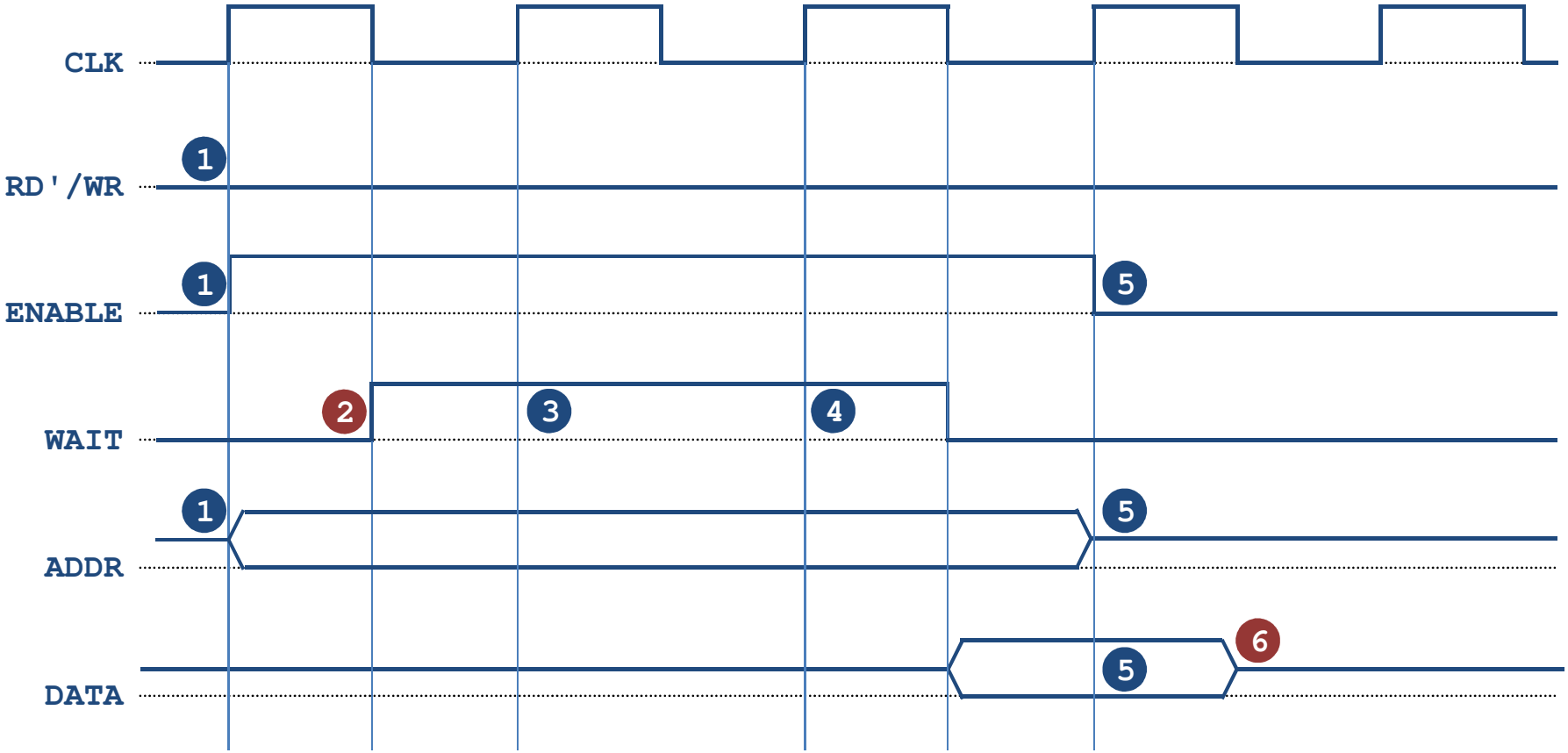


n Operazioni svolte dal bus master

k Operazioni svolte dal bus slave

Bus sincrono

- Ciclo di lettura con wait state

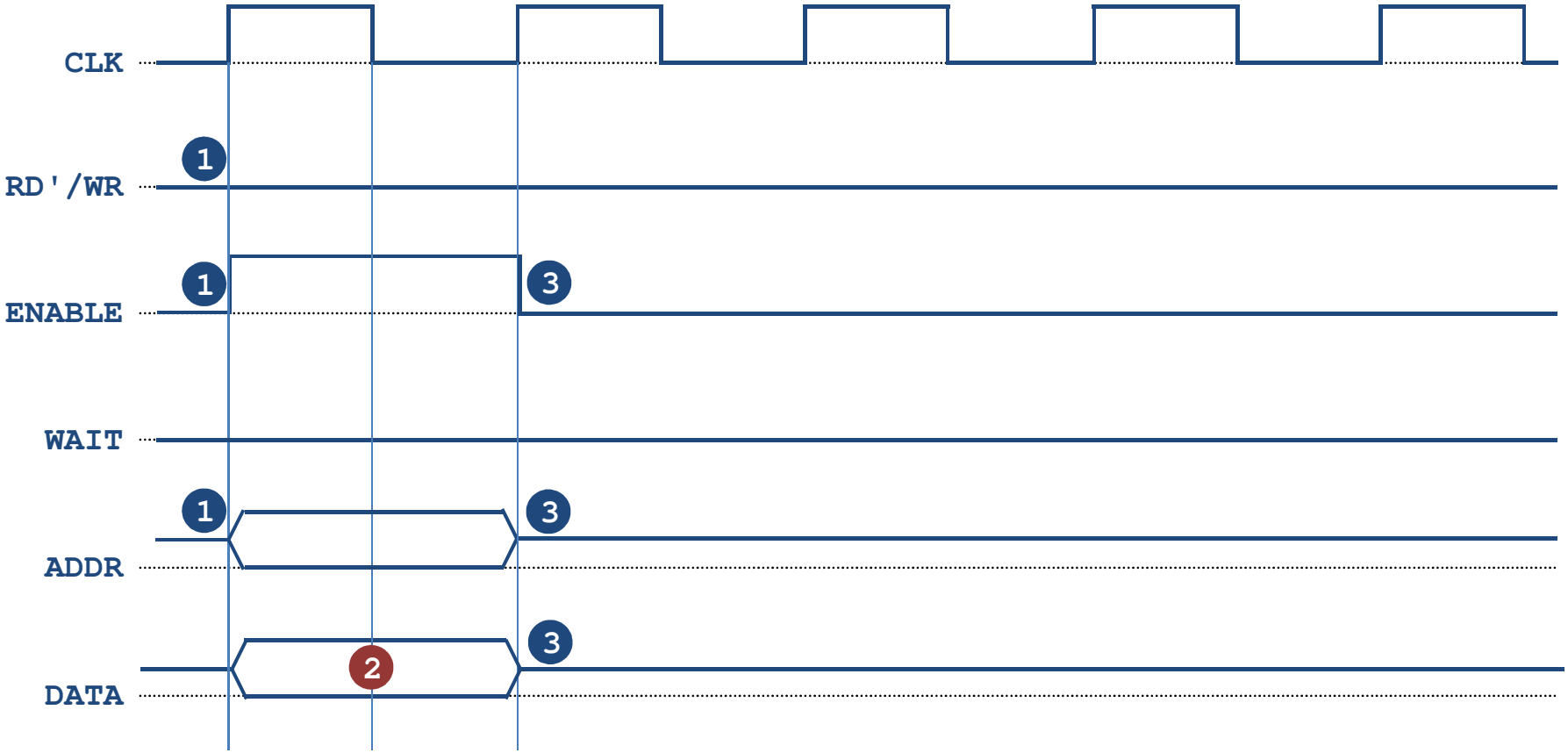


n Operazioni svolte dal bus master

k Operazioni svolte dal bus slave

Bus sincrono

- Ciclo di scrittura senza wait state

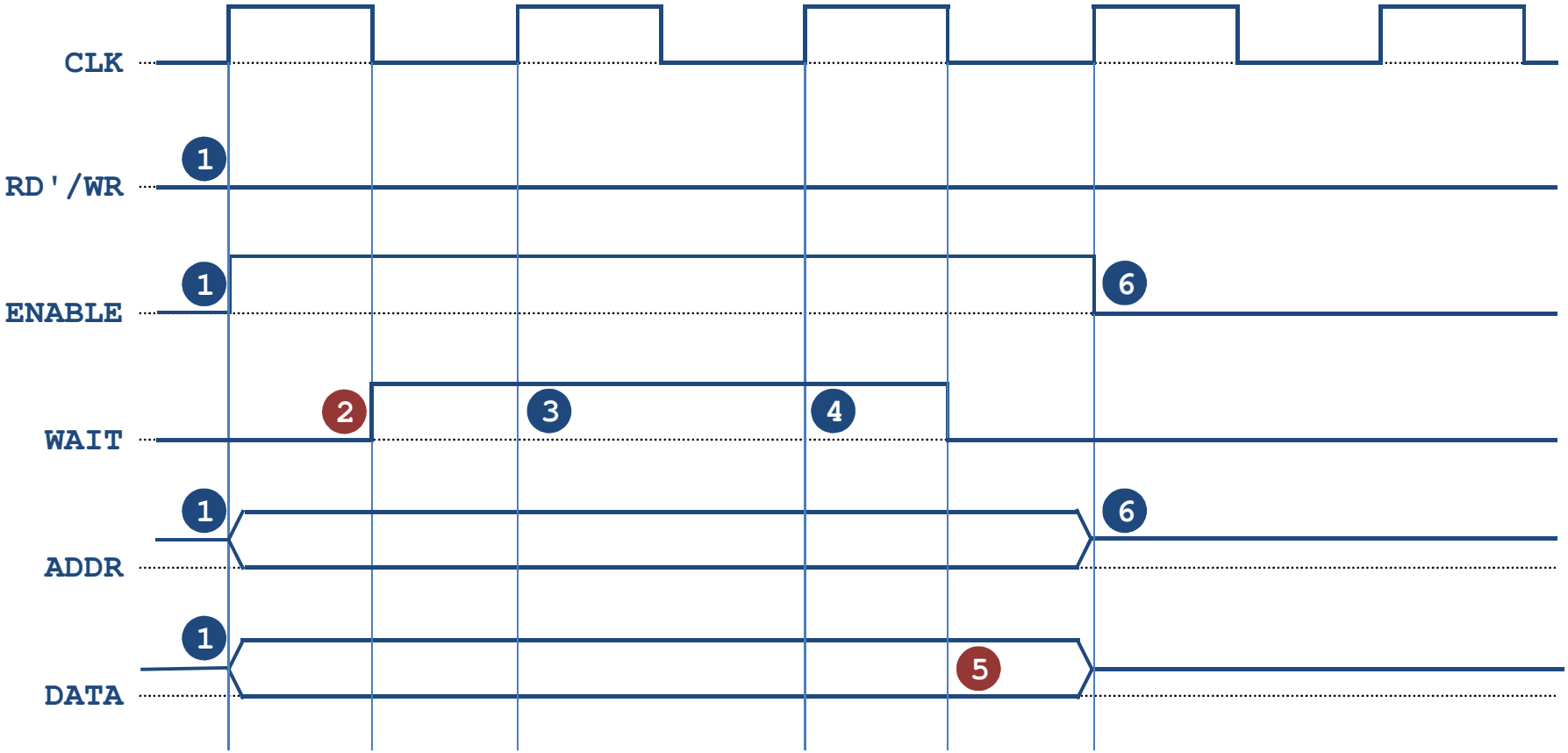


n Operazioni svolte dal bus master

k Operazioni svolte dal bus slave

Bus sincrono

- Ciclo di scrittura con wait state



n Operazioni svolte dal bus master

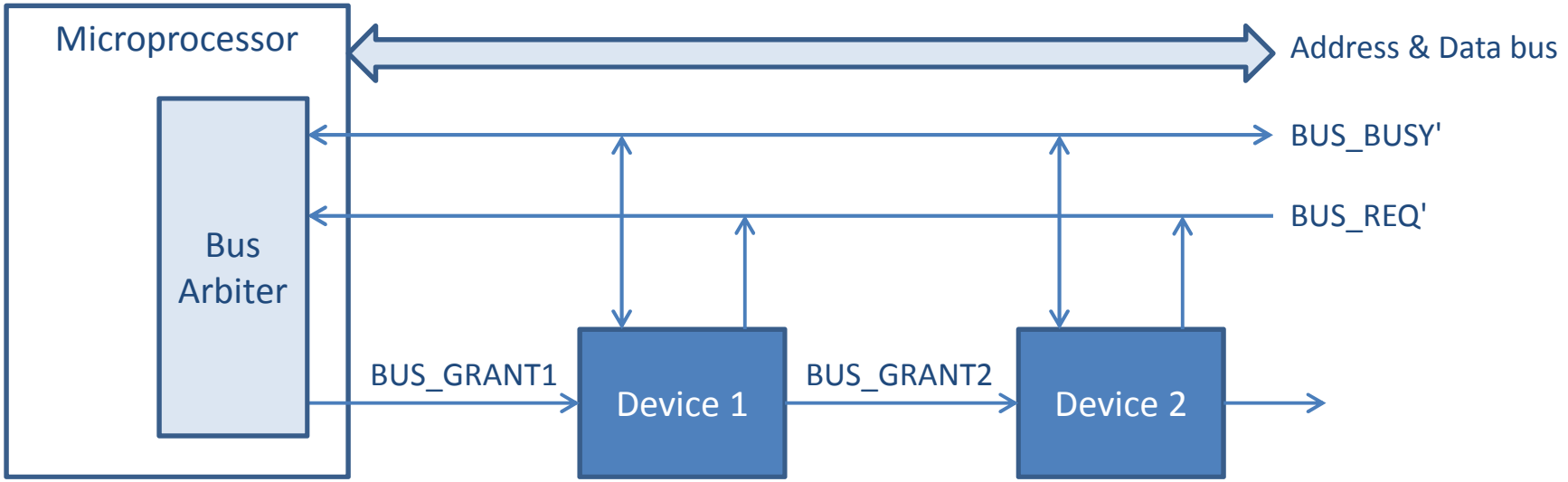
k Operazioni svolte dal bus slave

Arbitraggio

- **Negli schemi di comunicazione finora visti abbiamo identificato**
  - Un bus master
    - Unità che detiene il controllo del bus e che inizia le operazioni di trasferimento
  - Un bus slave
    - Unità che può unicamente rispondere alle richieste di trasferimento del master
- **Il ruolo bus master**
  - Non è necessariamente fissato
    - Una unità può essere sia master sia slave, in istanti di tempo diversi
    - In molti sistemi embedded il ruolo di bus master è fissato e definito al momento del progetto
  - Deve in generale essere definito mediante una forma di negoziazione
- **Chiamiamo "arbitraggio del bus" o "bus arbitration"**
  - La procedura di negoziazione con cui si stabilisce quale dispositvo è bus master, cioè detiene il controllo del bus, in un dato momento
- **Esistono due tipi di arbitraggio**
  - Arbitraggio centralizzato
  - Arbitraggio distribuito

Arbitraggio centralizzato

- **In questo schema una specifica unità si occupa dell'arbitraggio**
  - In genere si tratta di una unità del microprocessore
  - Può essere un circuito esterno dedicato al controllo
- **Lo schema di arbitraggio centralizzato prevede tre segnali di controllo**
  - BUS_REQ: Richiesta del bus
  - BUS_BUSY: Indica che il bus non è disponibile
  - BUS_GRANT: Assegnamento del bus

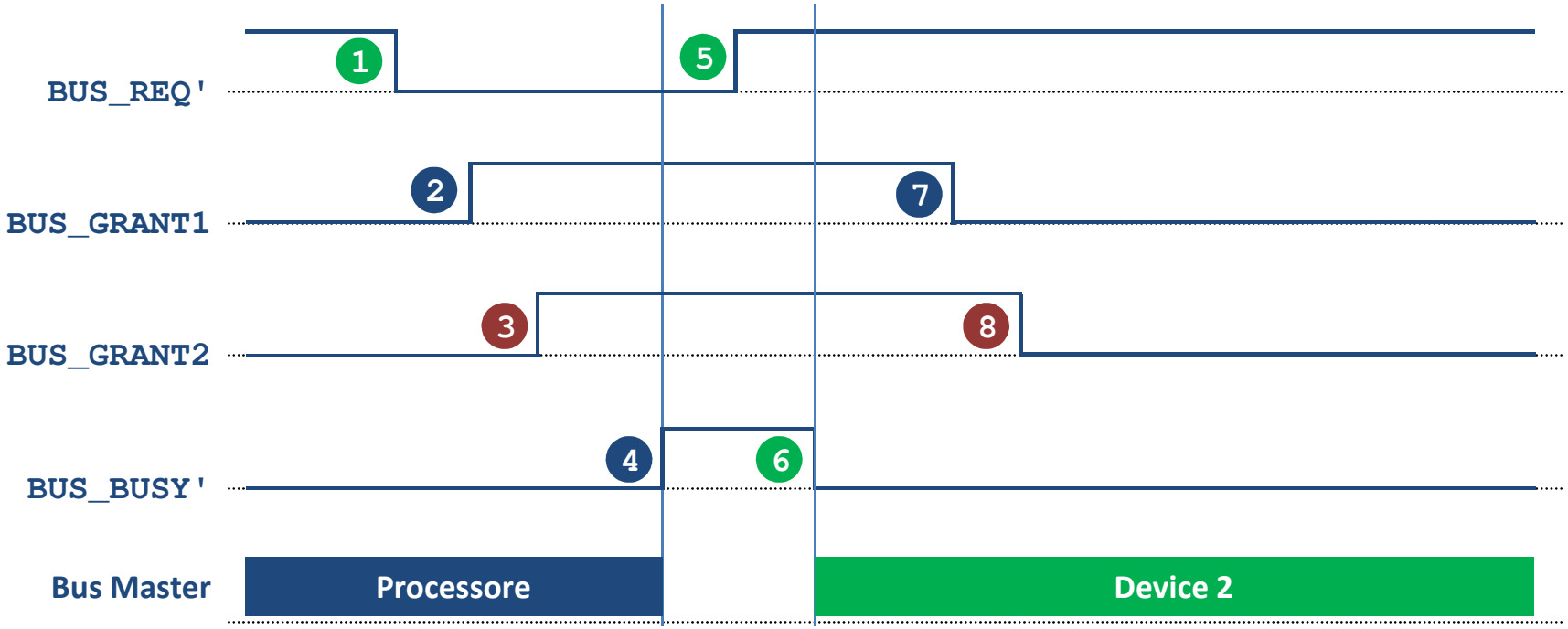


Arbitraggio centralizzato

- **Supponiamo che**
  - Il processore detenga il controllo del bus
  - il dispositivo DEVICE 2 voglia richiedere il controllo del bus
- **La sequenza di arbitraggio è la seguente**
  - Il DEVICE 2 porta a 0 la linea BUS_REQ
    - Si tratta di una linea open collector, come INT_REQ
    - Appena un device forza uno 0 sulla linea, essa va a 0
  - Appena il master (processore) vede la linea BUS_REQ a 0
    - Il master asserisce il segnale di GRANT
    - Dato che il bus grant è in daisy-chain, il processore asserisce la linea BUS_GRANT1
  - Il device propagano il segnale di grant
    - DEVICE 1, non essendo il dispositivo che ha fatto la richiesta, propaga il segnale al DEVICE 2
    - DEVICE 2, essendo il dispositivo che ha fatto la richiesta riceve ma non propaga il grant
    - Il DEVICE 2 sa che ha ottenuto il cotrollo del bus
  - La linea BUS_BUSY è in open collector e vale 0 se almeno un dispositivo detiene il bus
    - Pertanto DEVICE 2 deve attendere che la linea vada ad 1 per ottenere il bus
    - Non appena BUS_BUSY va ad 1, DEVICE 2 diviene bus master e forza nuovamente BUS_BUSY a 1

Arbitraggio centralizzato

- Sequenza di arbitraggio: il DEVICE 2 richiede il bus al processore



n Operazioni svolte dal processore

k Operazioni svolte dal device 1

k Operazioni svolte dal device 2

Arbitraggio distribuito

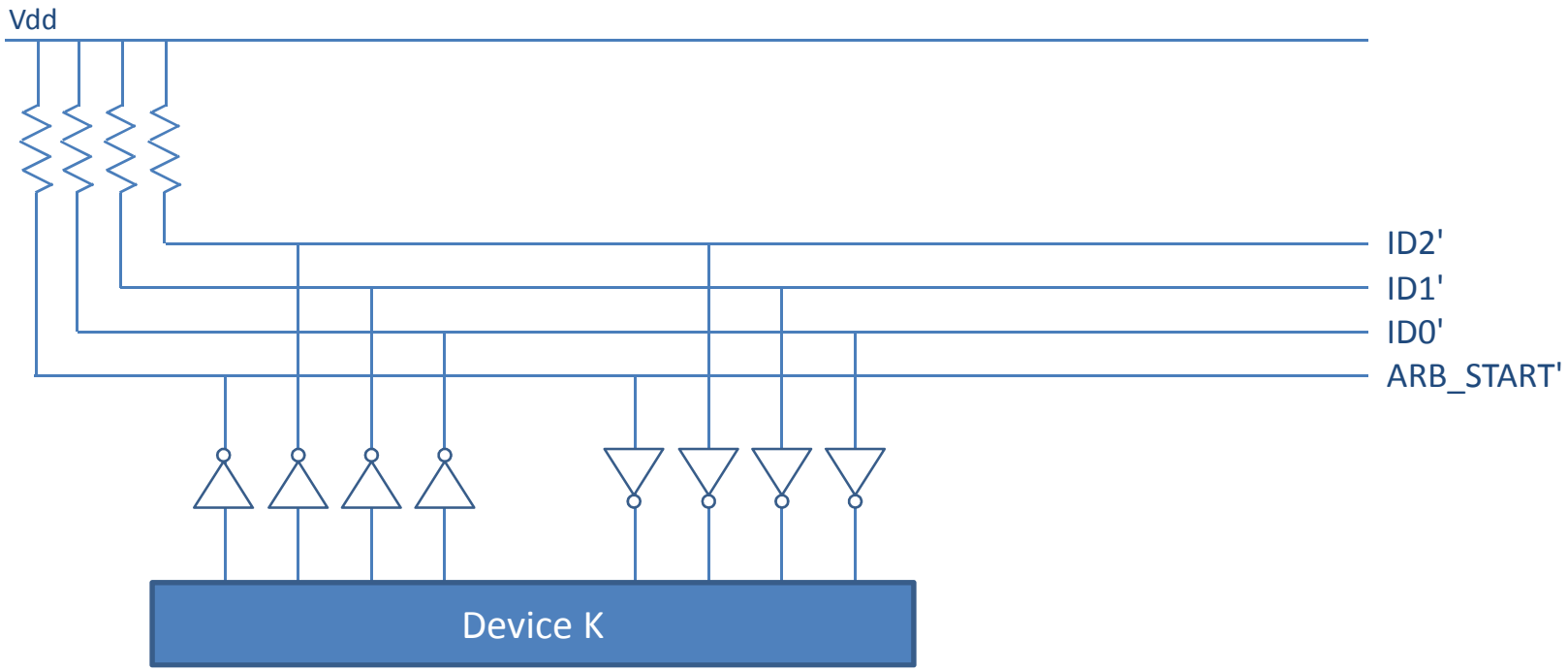
- **Nell'arbitraggio distribuito**
  - Tutte le unità partecipano al processo di negoziazione
  - Non esiste un arbitro del bus

- **Il sistema**
  - Richiede N linee di identificazione ID0, ID1, ...
    - Tutte le linee sono di tipo open collector
  - Richiede una linea di inizio arbitraggio ARB_START
    - Anche questa linea è open collector
  - Il numero di unità massimo è pari a 2^N

- **Ogni unità**
  - E' identificata da un codice (a volte chiamato indirizzo) ad N bit
  - Dispone di una interfaccia piuttosto complessa

Arbitraggio distribuito

- **L'interfaccia di una periferica che supporta l'arbitraggio distribuito**
  - Dispone di una sezione di uscita per la "scrittura" del proprio codice
  - Dispone di una sezione di ingresso per la "lettura" del codice presente sul bus
- **Lo schema di connessione è il seguente**



Arbitraggio distribuito

- **Le linee open collector sono tali per cui uno zero è forzante**
  - Si tratta dell'operazione di AND logico
  - Se più dispositivi richiedono il bus, il codice sulle linee ID è l'AND bit a bit dei codici
- **Supponiamo che due periferiche A e B richiedano il bus simultaneamente**
- **Sulle linee di codice giungono le richieste**
  - A = 101 ID(A) = 010
  - B = 110 ID(B) = 001
- **Le linee**
  - Assumono il valore ID(X) = (010) AND (001) = 000
  - Tale valore corrisponde a quello di una ipotetica periferica con codice X=111
- **Ogni dispositivo**
  - Confronta il codice rilevato sulle linee con il proprio
  - Esegue il confronto a partire dal bit più significativo
  - Non appena incontra una differenza
    - Smette di forzare uno 0 su quella linea e su tutte quelle meno significative

Arbitraggio distribuito

- **Nel nostro esempio**
  - A ha codice 101
  - A rileva il codice 111
- **La prima differenza si ha al bit in posizione 1**
  - La periferica smette di forzare uno zero sulle linee ID1 ed ID0
  - Il che equivale a dire che ora impone il codice 011
- **Sulle linee ora si ha**
  - A = [100] ID(A) = 011
  - B = 110 ID(B) = 001
- **Quindi ne risulta la configurazione**
  - ID(X) = (011) AND (001) = 001
- **A questo punto B rileva che le linee di codice corrispondono al proprio ID**
  - Infatti se sulle linee di codice si ha ID(X)=001 ilcodice è X = 110, cioè B
- **La periferica B detiene il controllo del bus**