Comunicazione di I/O

Daniele Paolo Scarpazza
Dipartimento di Elettronica e Informazione
Politecnico di Milano

30 Maggio 2005

Convenzioni sui termini e sui segnali

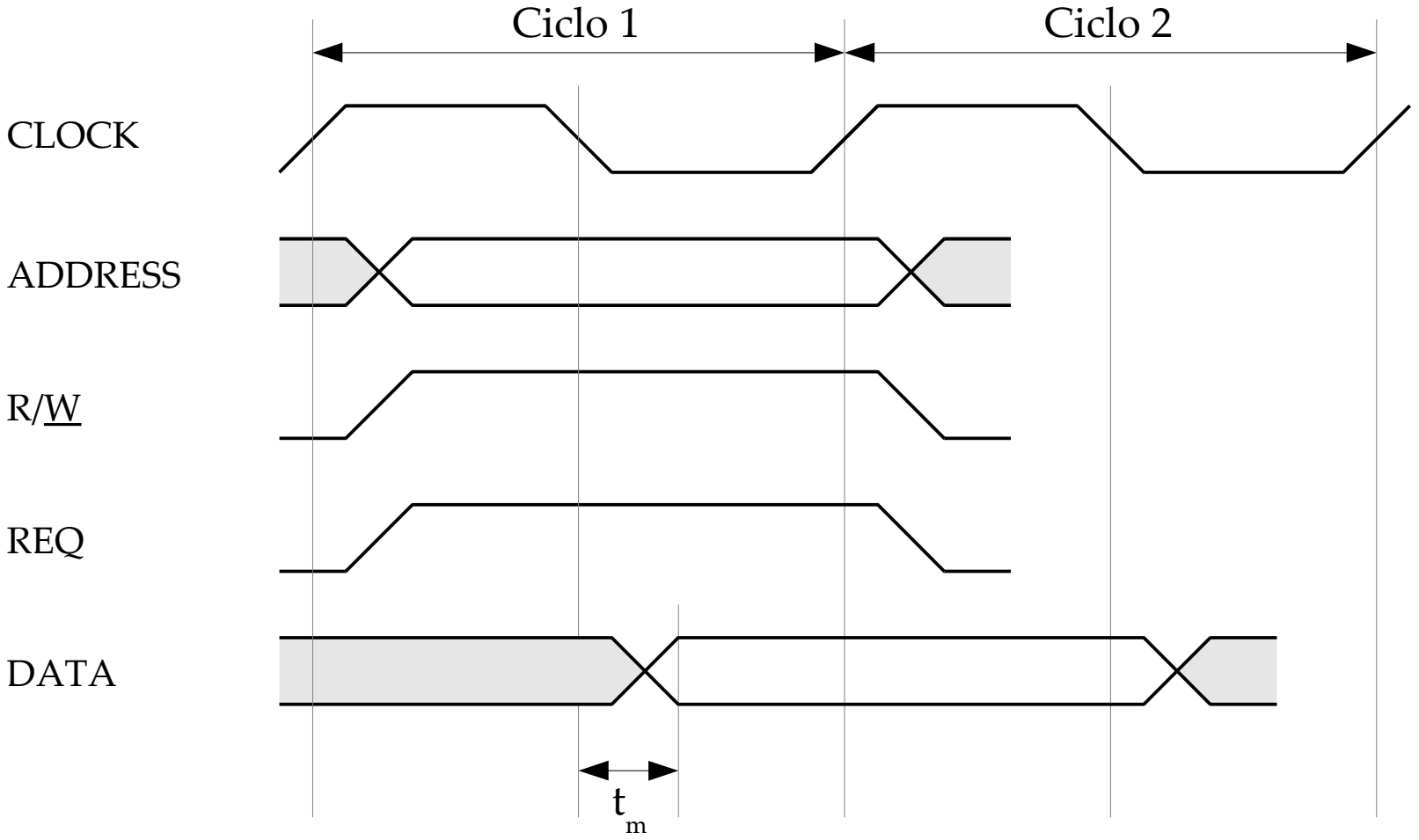
- Mostreremo sequenze degli eventi di meccanismi di I/O, con alcune precisazioni in più rispetto all'Hamacher, orientate allo svolgimento dei temi d'esame;
- Per convenzione il ciclo di clock inizia col fronte di **salita;**
- Convenzione di denominazione dei segnali (vedi Hamacher):
  - DATA = dato
  - ADDRESS = indirizzo
  - REQ = comando
  - SRDY = unità slave pronta
- I segnali attivi bassi sono qui sottolineati;
  Indipendentemente dal fatto che un segnale sia attivo alto/basso, usiamo i termini ***set*** / ***reset*** per indicare attivazione/disattivazione;
- Attenzione: Hamacher NON dice quando si fa unset di DATA; disambiguiamo noi in maniera conveniente (slides successive);

Bus sincrono monociclo, lettura

- Sequenza degli eventi:
  - @ clock ↗ master set ADDRESS, R/W
    master set REQ
  - @ clock ↘ slave sample ADDRESS, R/W
    ...
    slave set DATA
  - @ clock ↗ master sample DATA
    master unset REQ, ADDRESS
    (master può iniziare lettura successiva)
  - @ clock ↘ slave unset DATA
  - L'operazione successiva può essere una lettura
    (in tal caso si completa una lettura in ciascun ciclo);
    ma non può essere una scrittura: master deve attendere unset DATA;

Esempio: lettura monociclo

Clock = 10 MHz (T=100 ns); latenza della memoria t_m = 20ns; lettura in un ciclo;

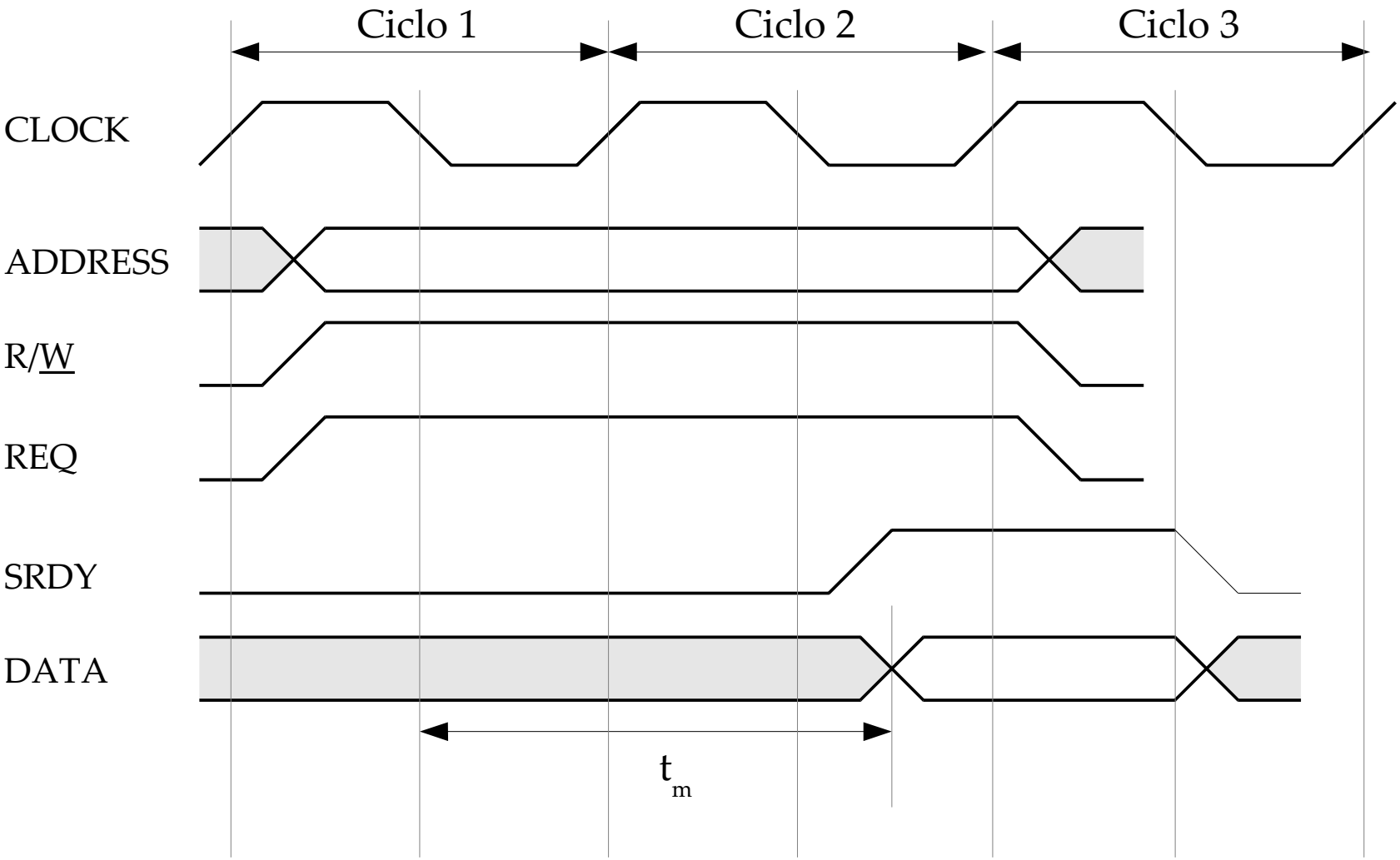


Bus sincrono multiciclo, lettura

- Sequenza degli eventi:
  - @ clock ↗ master set ADDRESS
    master set R/W
    master set REQ
  - @ clock ↘ slave sample ADDRESS, R/W
  - ritardo di 0 o più cicli di clock; a lettura completata:
    slave set SRDY
    slave set DATA
  - @ clock ↗ master sample DATA
    master unset REQ, ADDRESS
    (master può iniziare lettura successiva)
  - @ clock ↘ slave unset DATA, SRDY
- Il master campiona i dati al termine del ciclo in cui vede SRDY attivato;

Esempio: lettura multiciclo

Clock = 10 MHz (T=100 ns); latenza della memoria t_m = 120ns; lettura in 2 cicli;



Bus asincrono

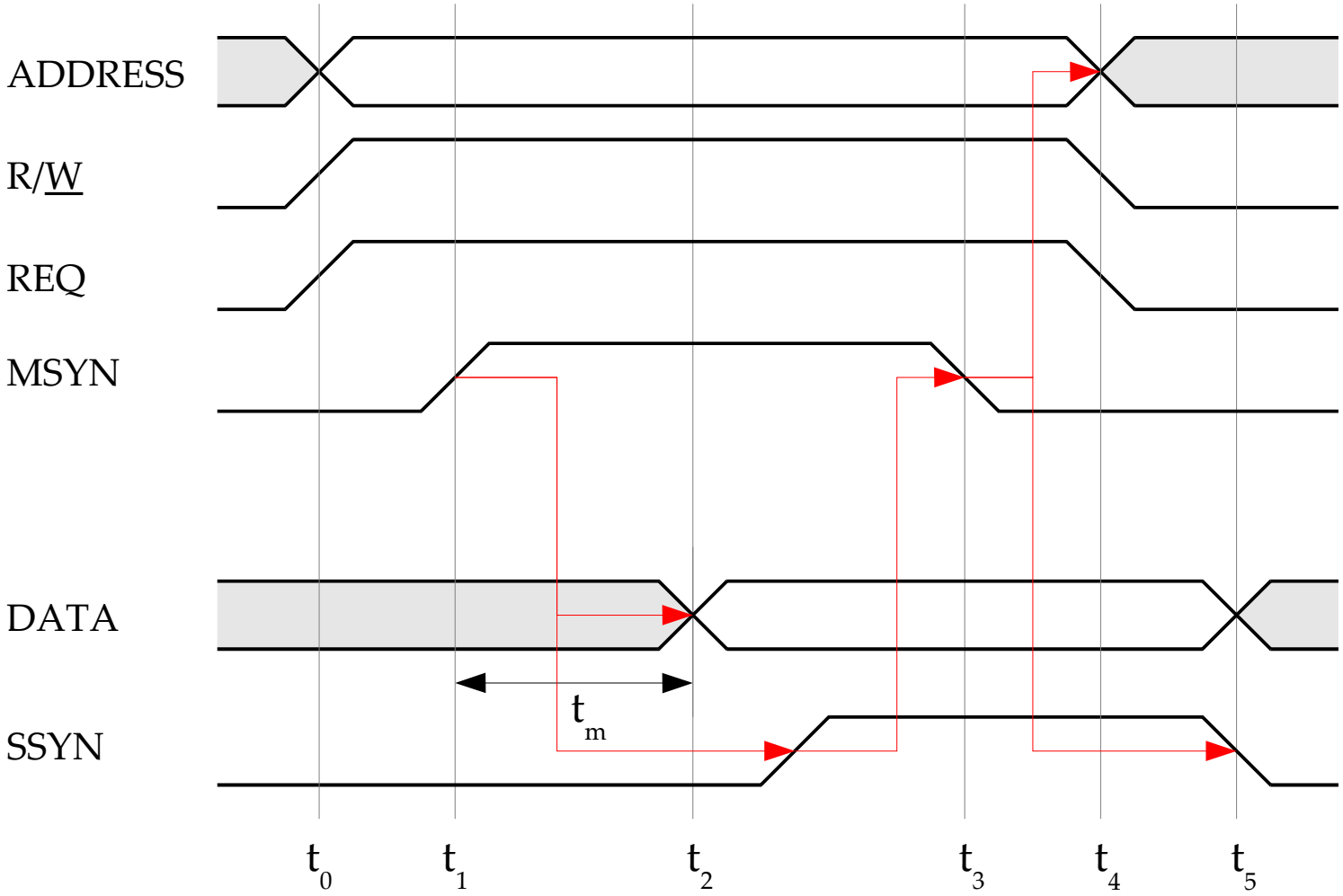
- Il bus asincrono non prevede segnale di clock;
- Le operazioni avvengono al massimo della velocità possibile, senza attendere fronti di clock;
- Al posto dei segnali di clock e SRDY, si usano i segnali:
  - MSYN, Master SYNchronization = unità master pronta;
  - SSYN, Slave SYNchronization = unità slave pronta;

Bus asincrono, lettura

- Sequenza degli eventi:
  - t0: master set ADDRESS, REQ, R/$\underline{W}$
  - t1: master set MSYN (dopo t prop.)
  - t2: slave set DATA (ASAP)
    slave set SSYN (dopo t prop.)
  - t3: master vede SSYN
    master sample DATA
  - t4: master reset MSYN, REQ, R/$\underline{W}$
  - t5: slave reset SSYN
    slave unset DATA

Bus asincrono, lettura

Diagramma temporale

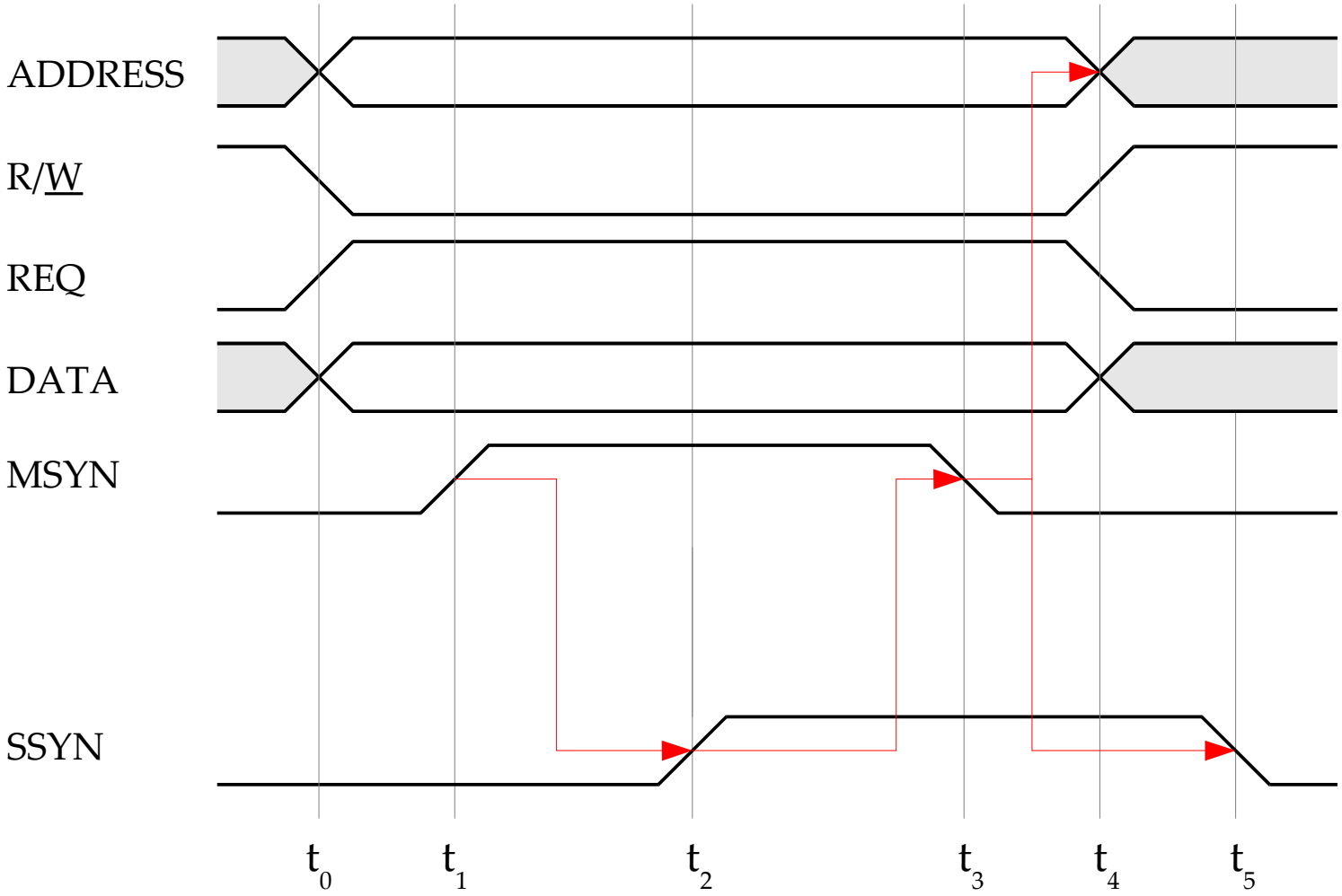


Bus asincrono, scrittura

- Sequenza degli eventi:
  - t0: master set ADDRESS, REQ, DATA
    master reset R/W (scrittura)
  - t1: master set MSYN (dopo t prop.)
  - slave sample DATA
    slave (scrittura)
  - t2: slave set SSYN
  - t3: master vede SSYN
  - t4: master unset MSYN, REQ, R/W, DATA
  - t5: slave reset SSYN

Bus asincrono, scrittura

Diagramma temporale



Arbitraggio del bus

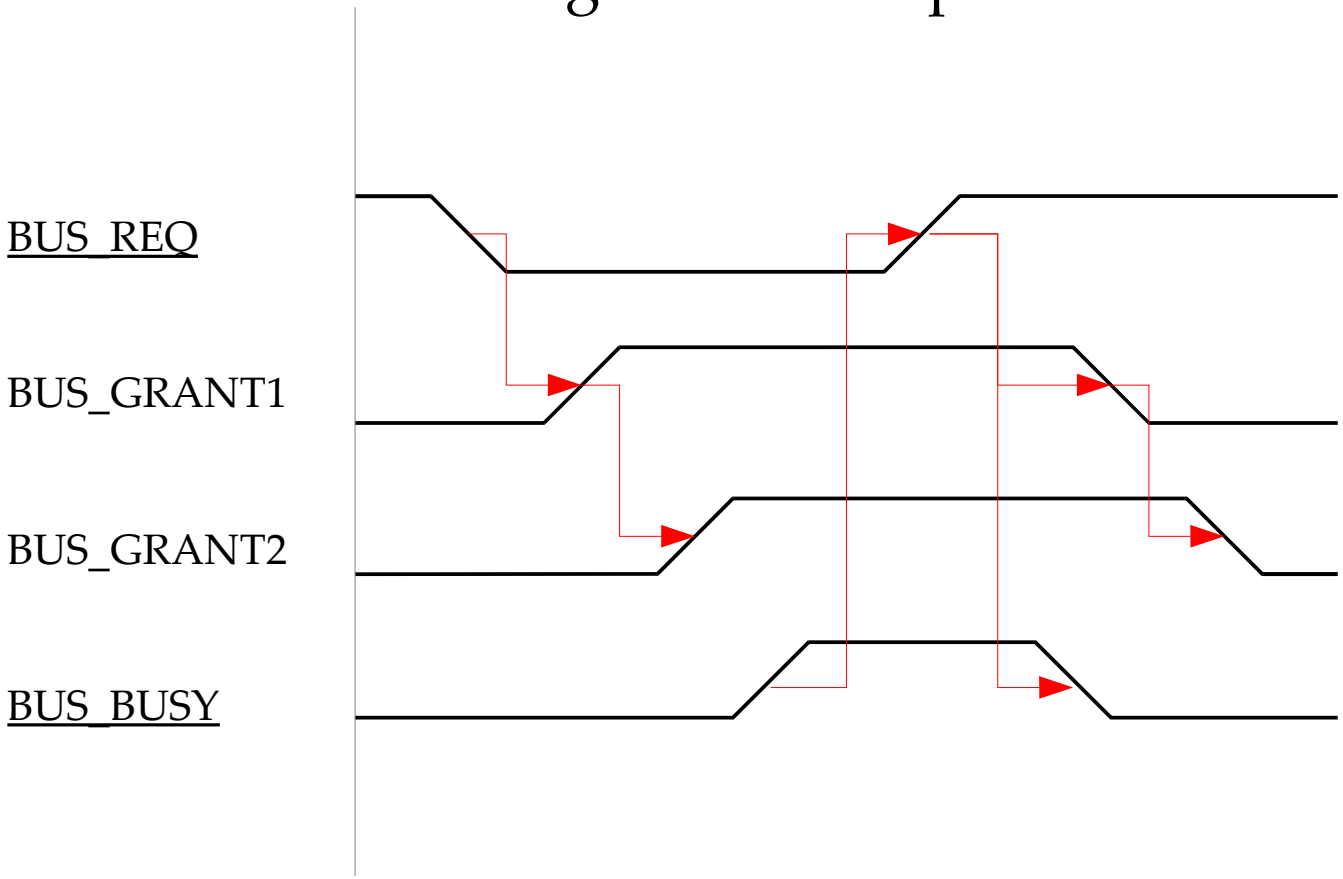
- DMA:
  - i dispositivi abilitati (*DMA controller*) comunicano con la memoria senza impegnare il processore;
  - ad un dato istante, uno solo dei DMA controller è il dispositivo abilitato a trasferire dati sul bus, cioè il *bus master;*
  - il passaggio dei poteri di bus master da un DMA controller ad un altro avviene:
    - in maniera centralizzata, sotto il controllo del *bus arbiter*, solitamente la CPU;
    - in maniera distribuita, senza bisogno di arbitri;

Arbitraggio del bus centralizzato

- Segnali:
  - Bus Request BUS_REQ richiesta del bus
  - Bus Granted BUS_GRANTx autorizzazione all'uso del bus
  - Bus BuSY BUS_BUSY bus occupato
- Sequenza degli eventi:
  - PERIF: set BUS_REQ
  - ARB: set BUS_GRANT1
  - PERIF: attende reset BUS_BUSY
    reset BUS_REQ
  - ARB: reset BUS_GRANT1
  - PERIF: set BUS_BUSY
    usa il bus
    reset BUS_BUSY
- Attenzione: numerose varianti possibili

Arbitraggio del bus centralizzato

Diagramma temporale



Interruzione vettorizzata

- Ogni dispositivo ha un suo codice univoco *n (vettore di interrupt)* e si identifica ponendo tale codice sul bus dati;
- In risposta all'IRQ, la CPU deve leggere *n* dal bus e saltare alla locazione indicata nella *n*-esima della della tabella dei vettori di interrupt;
- Il bus potrebbe essere in uso, nel momento in cui il dispositivo deve generare l'interrupt, quindi prima di porre il proprio codice sul bus attende il segnale INT_ACK (INTerrupt ACKnowlegde)
- A seconda dell'architettura adottata può esserci un segnale di WAIT per permettere al dispositivo di indicare alla CPU che il codice non è ancora pronto sul bus;

Interruzione vettorizzata

- Sequenza degli eventi:
  - PERIF: set INT_REQ
  - CPU: termina l'istruzione corrente
    set INT_ACK
  - PERIF: reset INT_REQ
    set WAIT se necessario
  - CPU: reset INT_ACK
  - PERIF: set DATA
    reset WAIT
  - CPU: attende reset WAIT
    sample DATA

Interruzione vettorizzata

Diagramma temporale