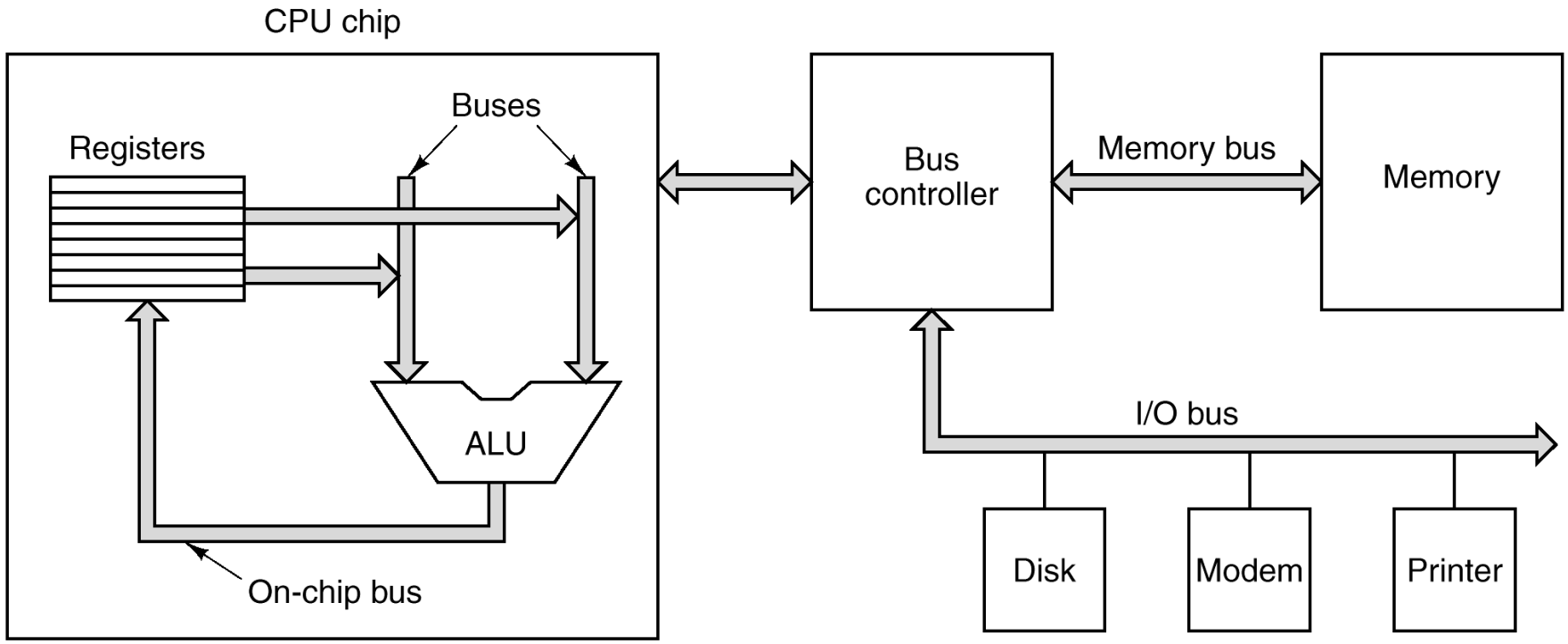
Calcolatori Elettronici

Parte V: Microprocessori e Bus

Prof. Riccardo Torlone

Universita di Roma Tre

Architettura a più Bus



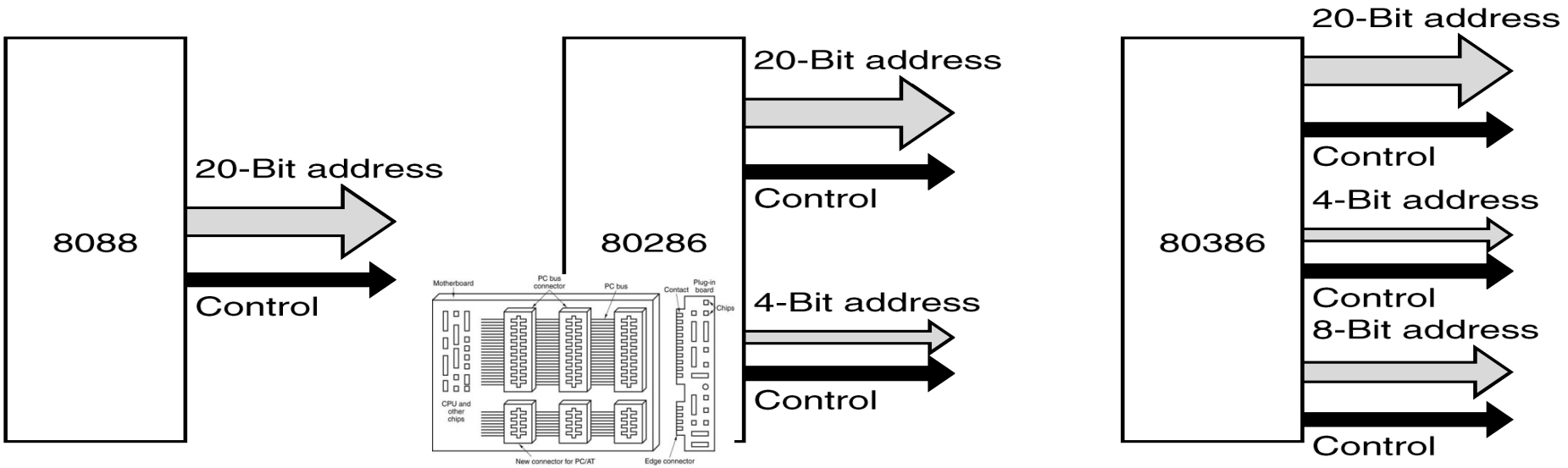
- Diversi bus, interni ed esterni al chip
- Soddisfano diverse esigenze:
  - Velocità di trasferimento
  - Numero di linee
  - Compatibilità all'indietro
- Negli attuali PC almeno due bus esterni

Comunicazione sul Bus

Master	Slave	Example
CPU	Memory	Fetching instructions and data
CPU	I/O device	Initiating data transfer
CPU	Coprocessor	CPU handing instruction off to coprocessor
I/O	Memory	DMA (Direct Memory Access)
Coprocessor	CPU	Coprocessor fetching operands from CPU

- La comunicazione sul bus è regolata da un *protocollo* di bus
- In ciascun ciclo comunicano due soli dispositivi il **master** e lo **slave**
- Lo stesso dispositivo può avere ruoli diversi a seconda dei casi
- I dispositivi sono connessi al bus tramite un *bus transceiver*
- La connessione al bus o avviene tramite dispositivi a tre stati oppure è di tipo *open collector*

"Larghezza" del Bus



- Larghezza = numero di linee
- Linee indirizzo: dimensione dello spazio (di memoria) indirizzabile, 2^n locazioni con *n* bit di indirizzo
- Linee dati + velocità di trasmissione: banda di trasferimento
- Condivisione di più segnali sulla stessa linea per diminuire i costi
- **Problema:** al crescere della velocità del bus aumenta il *bus skew* (differenza nella velocità di propagazione dei segnali su linee diverse)

Segnali asseriti e negati

In alcuni casi (a seconda delle scelte di progetto) un segnale provoca l'azione corrispondente quando la sua tensione è alta (1), in altri quando è bassa (0).

Per evitare confusione si parla di:

- Segnale asserito: quando assume il valore (alto o basso) che provoca l'azione
- Segnale negato: altrimenti

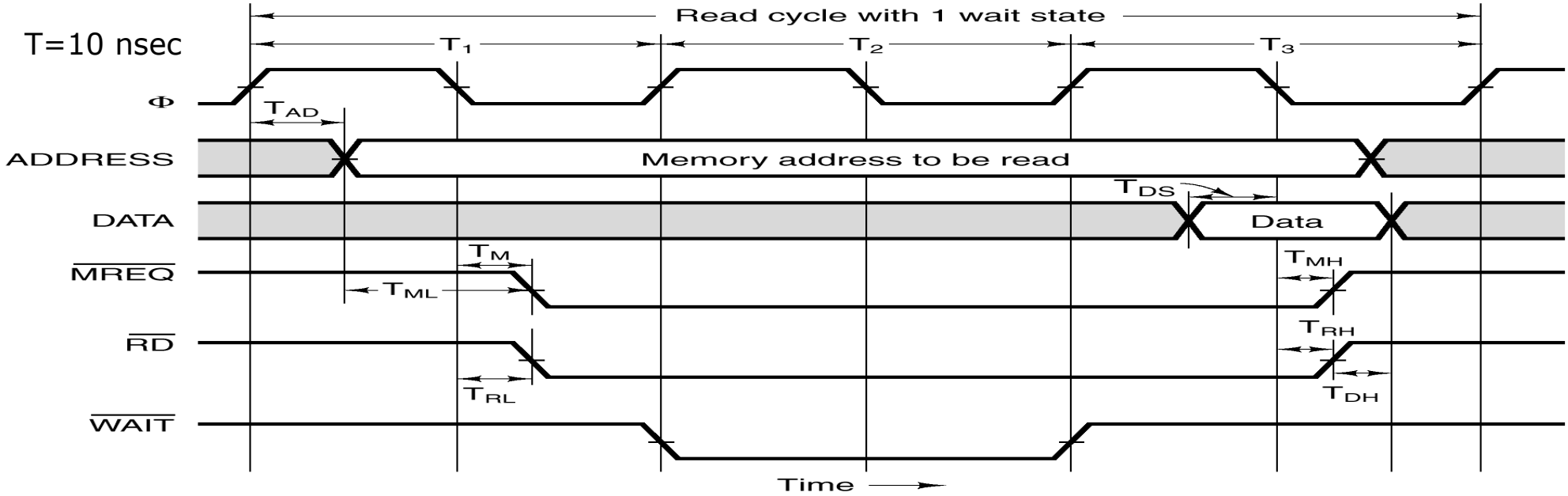
Si adotta la seguente notazione:

- S: segnale che è asserito alto
- $\overline{S}$: segnale che è asserito basso

Ulteriore notazione (usata da Intel):

- S: segnale che è asserito alto
- S#: segnale che è asserito basso
  - (adatta al set di caratteri ASCII)

Bus Sincroni: ciclo di lettura



Symbol	Parameter	Min	Max	Unit
T_{AD}	Address output delay		4	nsec
T_{ML}	Address stable prior to $\overline{MREQ}$	2		nsec
T_M	$\overline{MREQ}$ delay from falling edge of Φ in T_1		3	nsec
T_{RL}	RD delay from falling edge of Φ in T_1		3	nsec
T_{DS}	Data setup time prior to falling edge of Φ	2		nsec
T_{MH}	$\overline{MREQ}$ delay from falling edge of Φ in T_3		3	nsec
T_{RH}	$\overline{RD}$ delay from falling edge of Φ in T_3		3	nsec
T_{DH}	Data hold time from negation of $\overline{RD}$	0		nsec

(b)

Bus Sincrono: Temporizzazione

ES

Frequenza 100 MHz, periodo 10 nsec.

- primo vincolo: tempo a disposizione della memoria fra:
  a) la comparsa dell'indirizzo sul Bus
  b) la disponibilità dei dati sul Bus

$$\tau_1 = 2.5 \times T - T_{AD} - T_{DS} = 25 - 4 - 2 = 19 \text{ nsec}$$

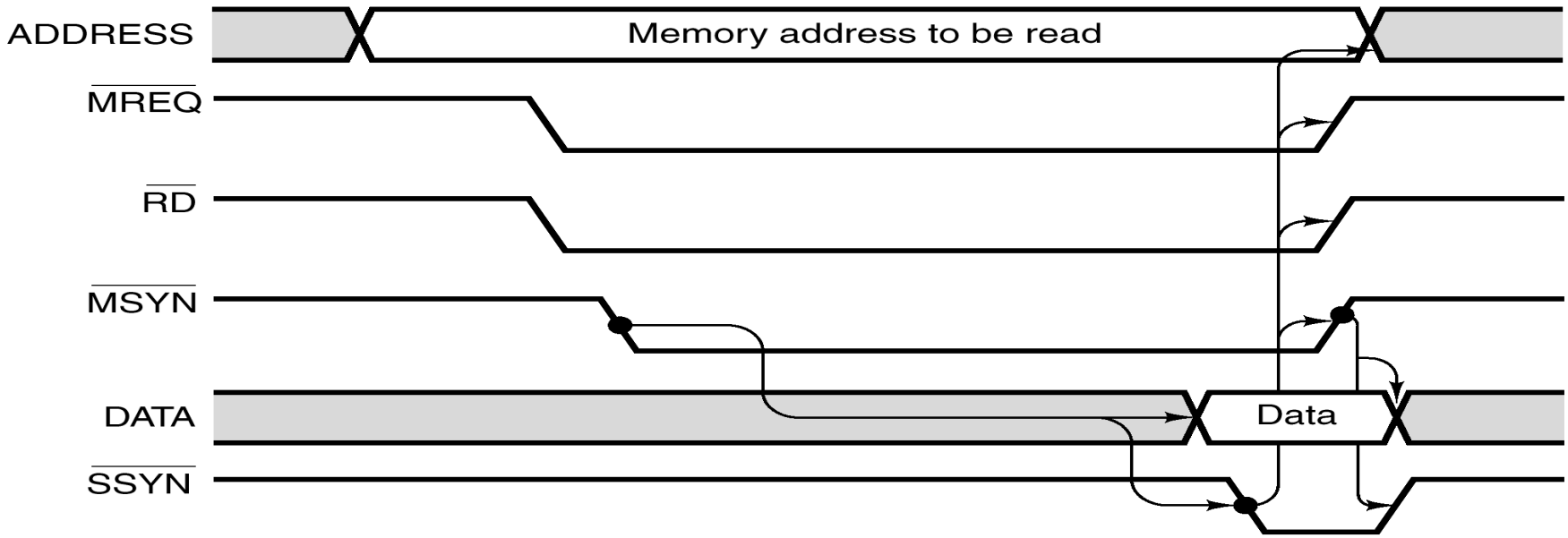
(una memoria da 10 nsec ce la fa di sicuro)

- secondo vincolo: tempo a disposizione della memoria fra:
  a) l'asserzione di $\overline{\text{MREQ}}$ e $\overline{\text{RD}}$
  b) la disponibilità dei dati sul Bus

$$\tau_2 = 2 \times T - T_M - T_{DS} = 20 - 3 - 2 = 15 \text{ nsec}$$

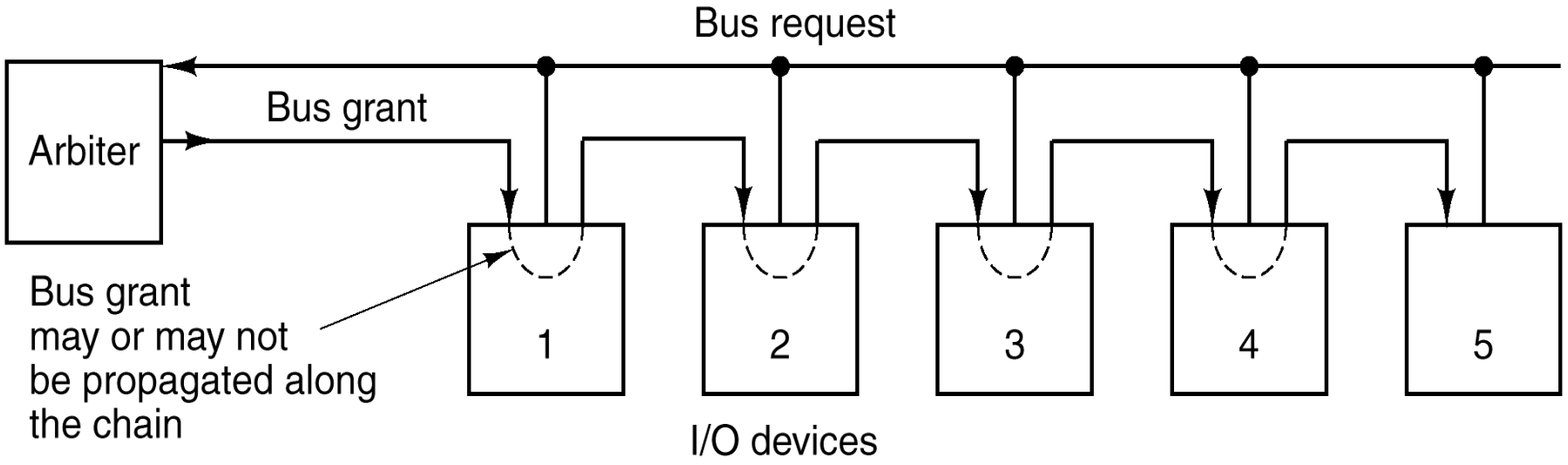
Se il chip di memoria non soddisfa questi requisiti mantiene asserito il segnale di $\overline{\text{WAIT}}$ per introdurre *stati di wait*, cioè cicli di bus addizionali.

Bus Asincrono: ciclo di lettura



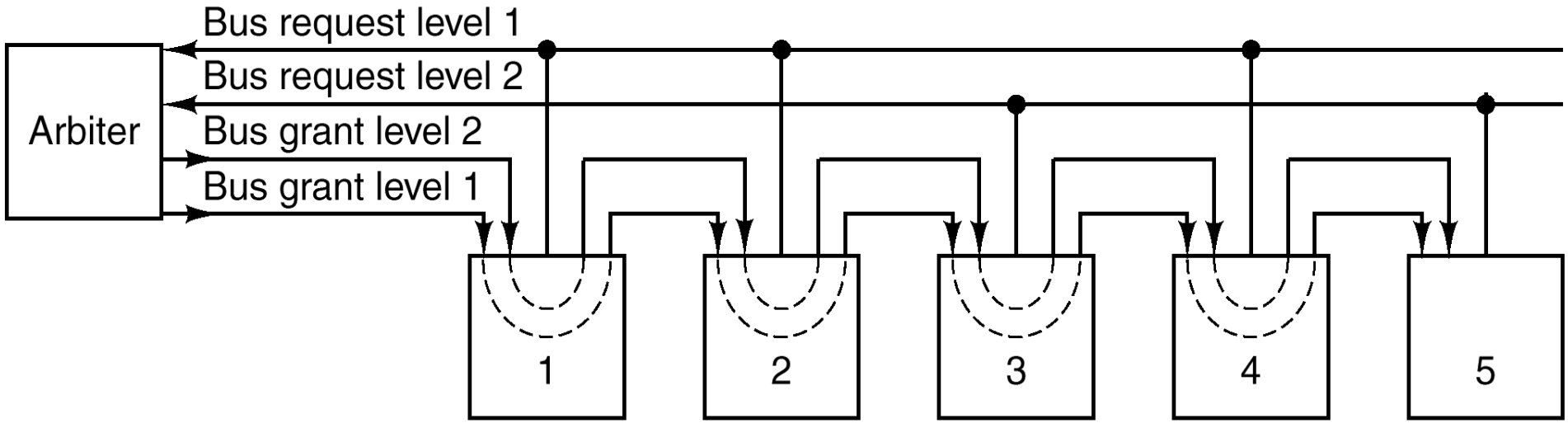
- Accoppiamento di dispositivi con velocità diverse
- Gli eventi avvengono in risposta ad altri eventi
- **Full handshake**:
  a) $\overline{\text{MSYN}}$ asserito
  b) $\overline{\text{SSYN}}$ asserito in risposta a $\overline{\text{MSYN}}$
  c) $\overline{\text{MSYN}}$ negato in risposta a $\overline{\text{SSYN}}$
  d) $\overline{\text{SSYN}}$ negato in risposta alla negazione di $\overline{\text{MSYN}}$

Arbitraggio del Bus



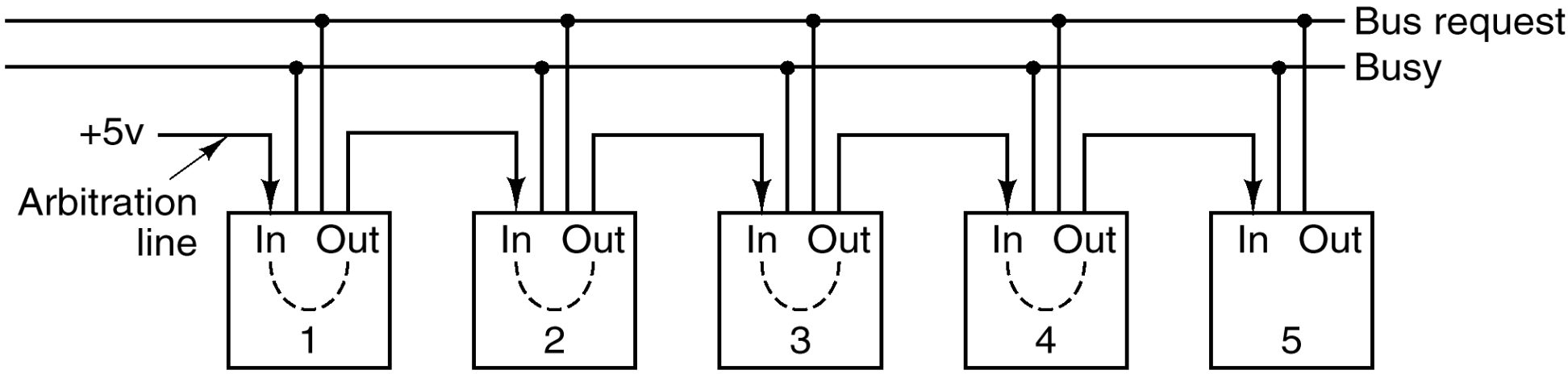
- Permette di decidere quale dispositivo sarà il prossimo Bus Master risolvendo eventuali conflitti
- Spesso l'arbitro è nel chip del microprocessore
- Linea di richiesta condivisa
- Il *Bus grant* è propagato dall'arbitro prima dell'inizio del ciclo
- Viene intercettato dal futuro master
- NB: Favoriti i dispositivi situati vicino all'arbitro

Livelli Multipli di priorità



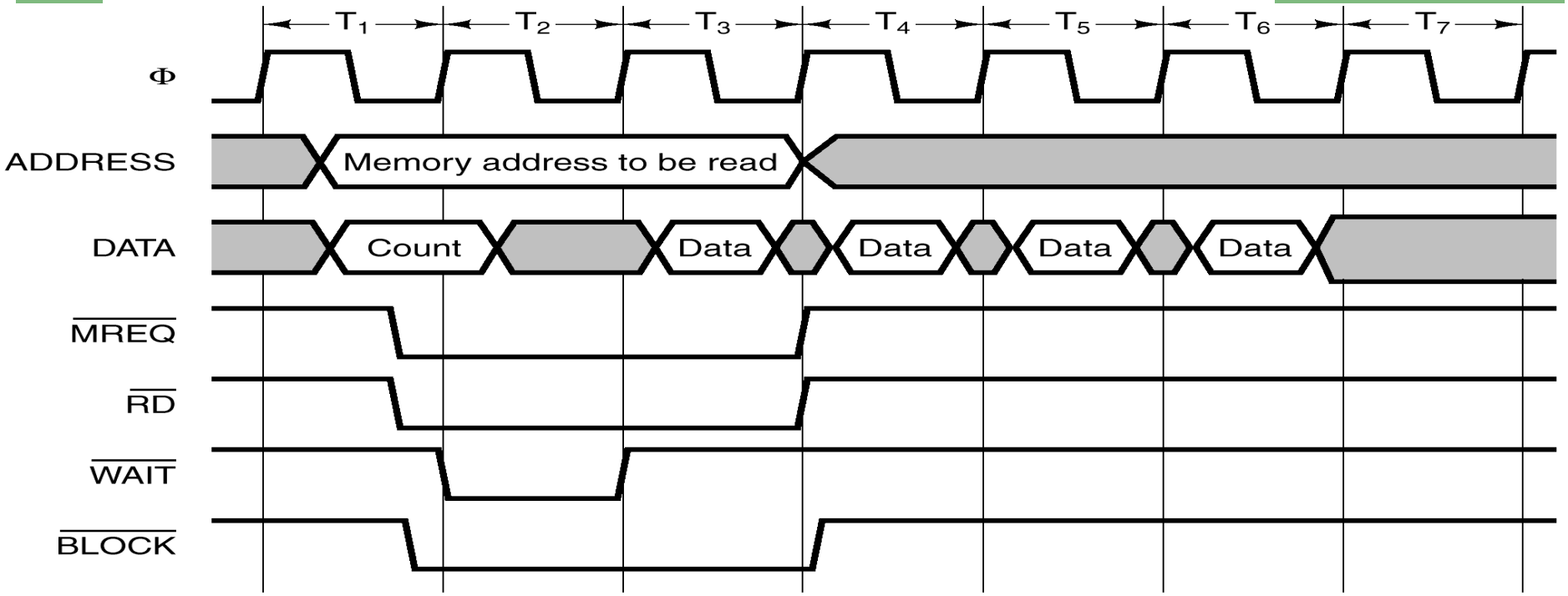
- Diverse linee di richiesta associate a diversi *livelli di priorità*
- In caso di conflitto favorite le catene a priorità più alta
- All'interno di ciascuna catena vale la posizione
- In genere se c'è un solo bus con anche la memoria, la CPU ha priorità più bassa dei dispositivi di I/O (e.g. dischi)

Arbitraggio Decentralizzato



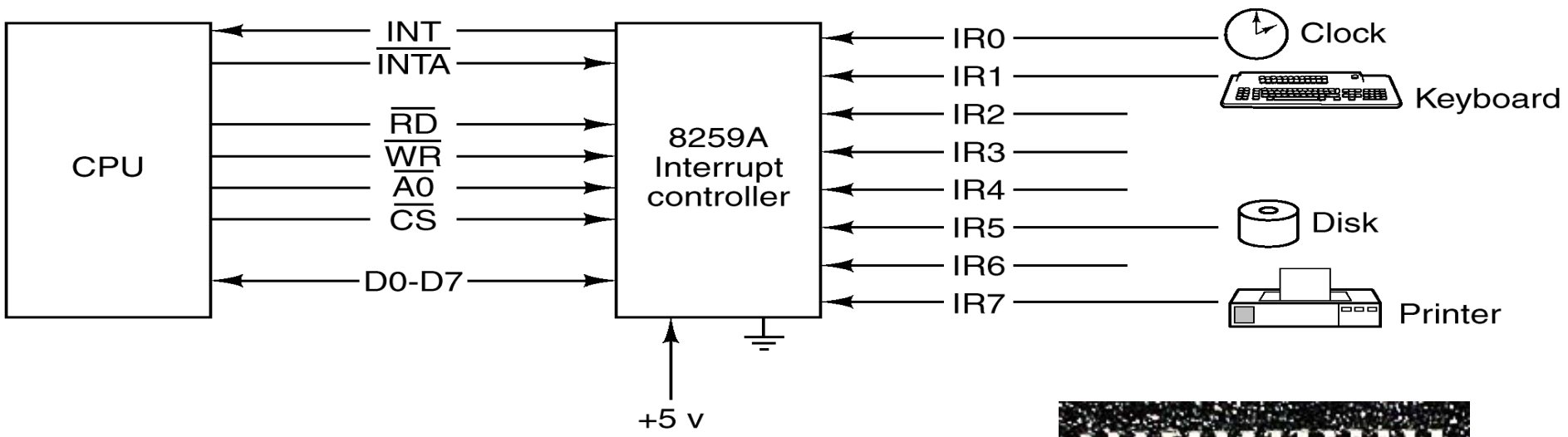
- Quando nessun dispositivo vuole il Bus, la linea di arbitraggio è asserita con propagazione a tutti i dispositivi
- Quando un dispositivo vuole il Bus:
  - invia una richiesta di bus
  - verifica se il bus è libero
  - se **In** è asserito diventa master, nega **Out** e asserisce **Busy**
  - se **In** non è asserito non diventa master e nega **Out**
- Non necessita di arbitro, è più semplice e più veloce

Block Transfers



- Usato per trasferire *blocchi* di dati
- Numero di parole specificato durante T_1
- Dopo la prima viene trasferita una parola ogni ciclo (invece di una ogni tre cicli)
- Per leggere quattro parole occorrono 6 cicli invece di 12
- Il segnale $\overline{\text{BLOCK}}$ viene asserito per chiedere un block transfer

Gestione delle Interruzioni

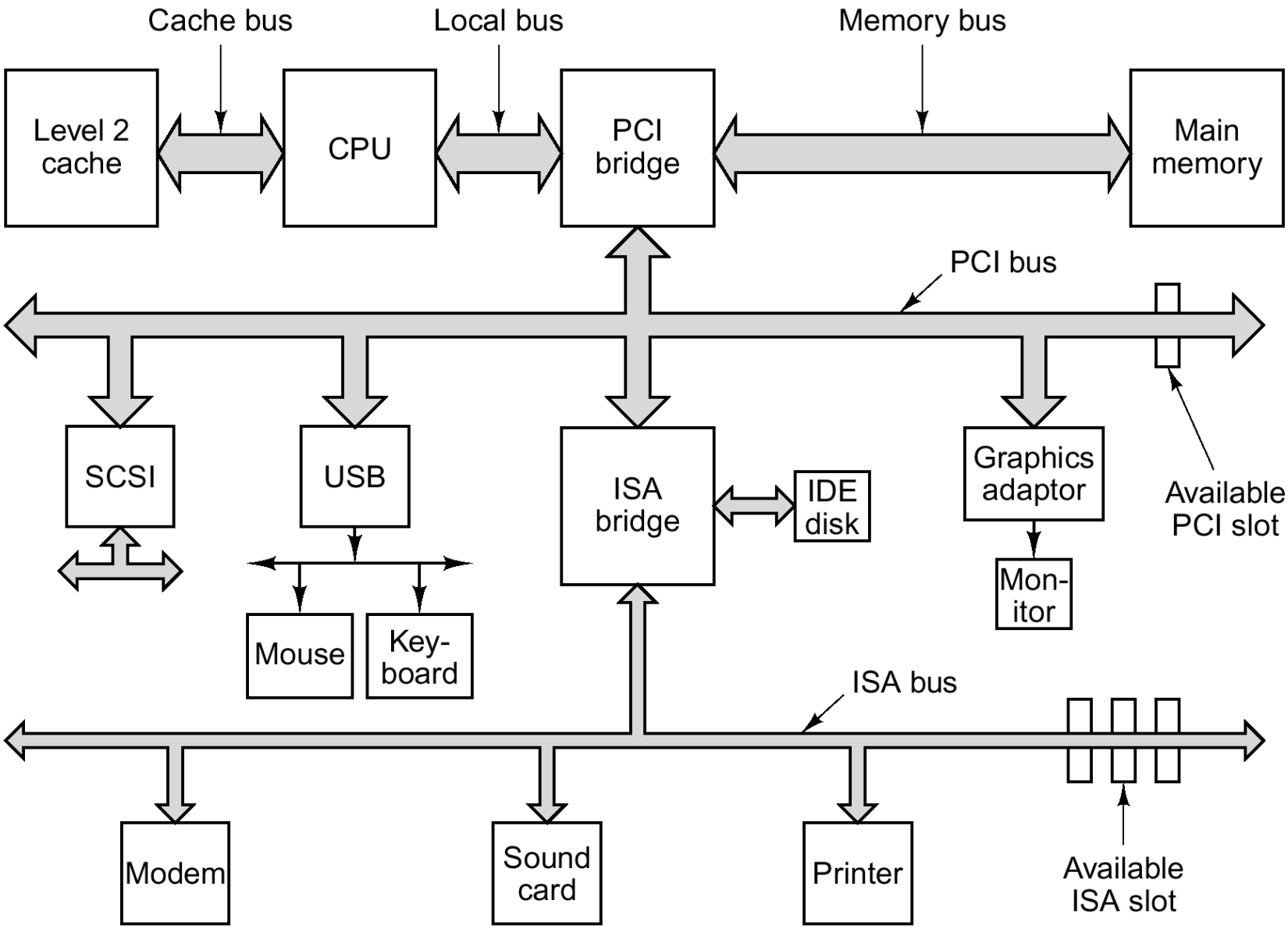


- Chip controllore di interruzioni
- Gestisce 8 linee di interrupt
  - INT: interruzione inviata alla CPU
  - $\overline{\text{INTA}}$: *acknowledge* della CPU
  - Numero del dispositivo sul bus
  - Il numero viene usato come indice di un *vettore di interrupt*
  - IR0-IR7: linee di interrupt
- Il vettore di interrupt è usato dalla CPU per invocare la relativa procedura
- Registri all'interno del chip scrivibili dalla CPU per programmare il controller

Bus reali

- Requisiti:
  - Video a 1024×768 con 3B per pixel (true color)
    - 1 frame: 2.25 MB
    - 30 frame al secondo: 67.5 MB/sec
    - HD->RAM->VRAM: 135 MB/sec
  - Video a 1920×1080 con 3B per pixel
    - 1 frame: ~5.2 MB
    - 30 frame al secondo: 155 MB/sec
    - HD->RAM->VRAM: 310 MB/sec
- Bus legacy:
  - ISA: 8.33Mhz, 2B per ciclo, 16.7MB/sec
  - EISA: 8.33Mhz, 4B per ciclo, 33.3MB/sec
- Bus PCI (1990, Intel): fino a 528 MB/sec
- Bus AGP (fine anni ’90): fino a 2.1 GB/sec
- Bus PCIe (2004, Intel/Dell/HP/IBM): 16GB/sec e oltre

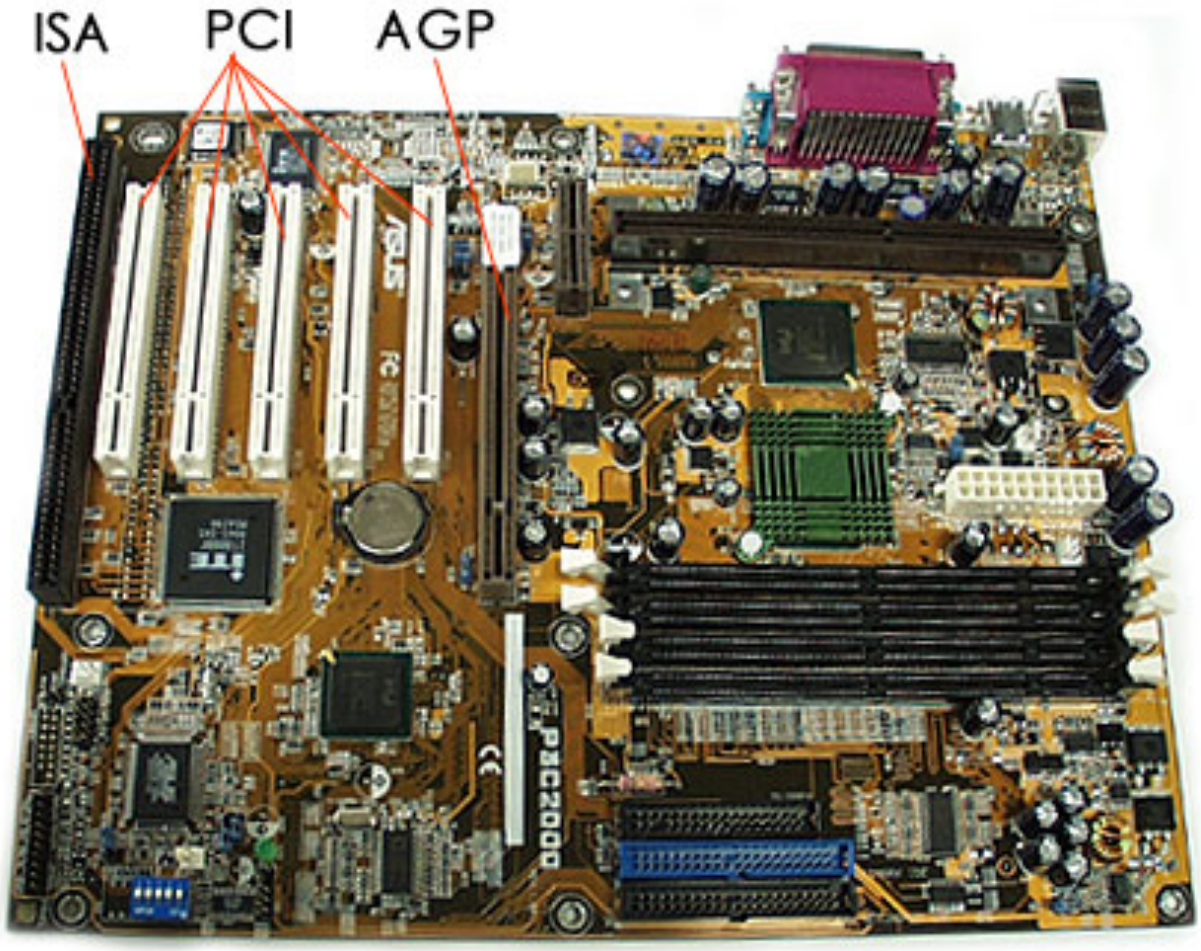
Il Bus PCI: architettura originale

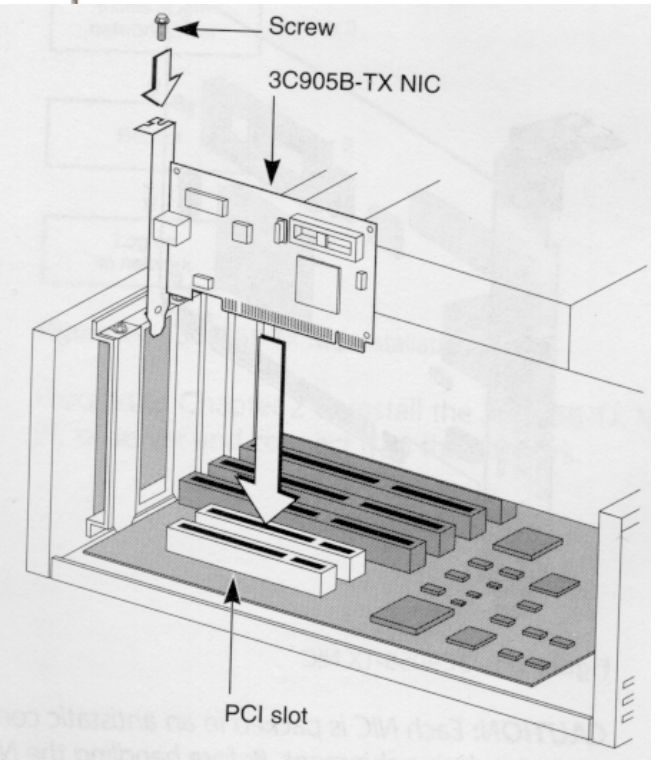


BUS AGP

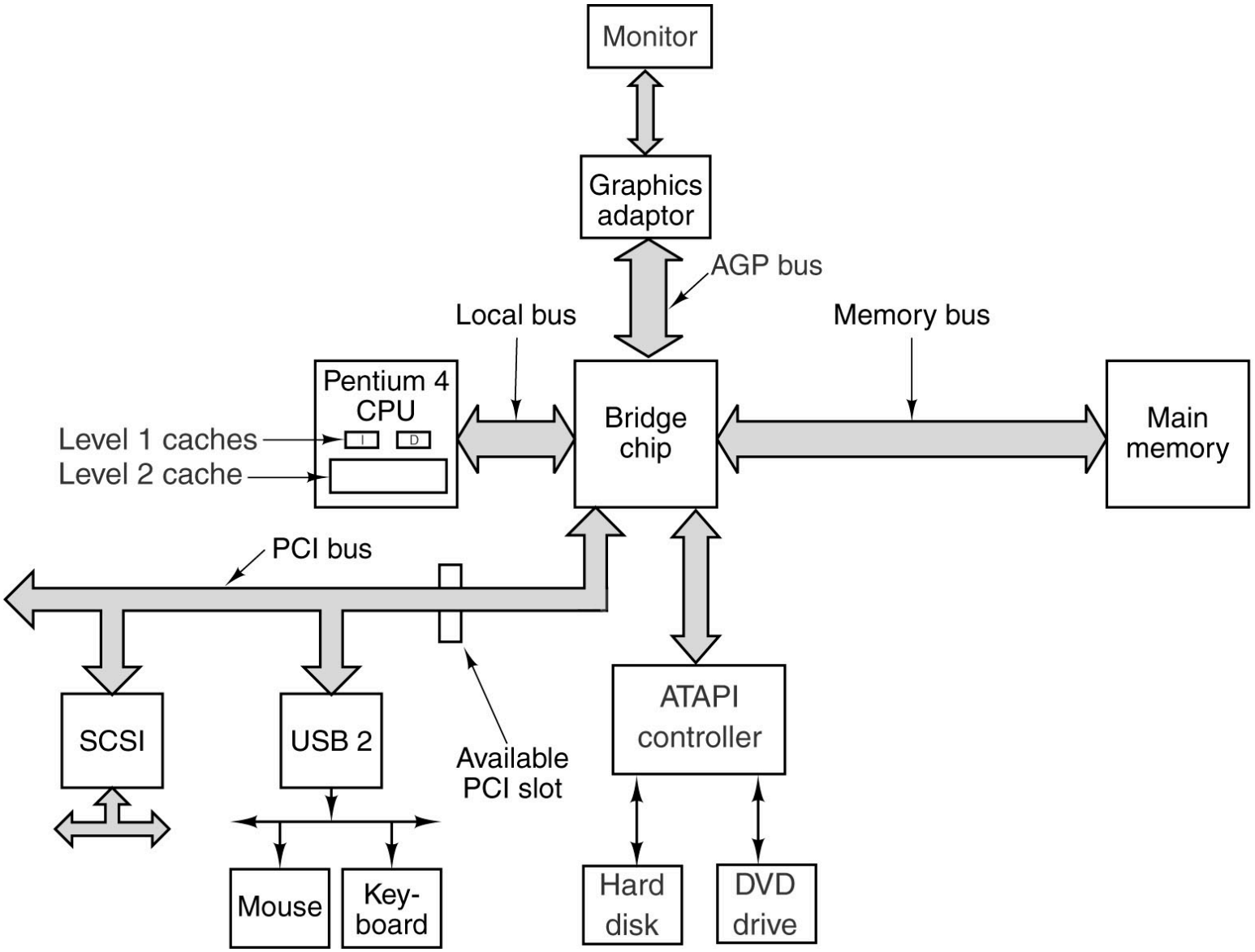
- AGP (Accelerated Graphic Port): bus espressamente dedicato per le schede grafiche;
- Il bus prevede la presenza di un nuovo slot con un connettore diverso da quello PCI sulla scheda madre;
- Il protocollo prevede la presenza di un solo dispositivo;
- A differenza del PCI, adotta una tecnica di pipeling;
- Funzionamento:
  - modalità 1x (AGP 1.0): 66 MHz, banda 264 Mbyte/s.
  - modalità 2x: sfrutta sia il fronte di salita del ciclo di clock, che quella di discesa; frequenza virtuale di 133 MHz e banda di 532 Mbyte/s.
  - modalità 8x (AGP 3.0): 2.1 GB/sec
- Le schede AGP adottano il sistema *Direct Memory Execute* (DIME) che consente al controller di svolgere elaborazioni usando la memoria principale

Schede e slot PCI e AGP



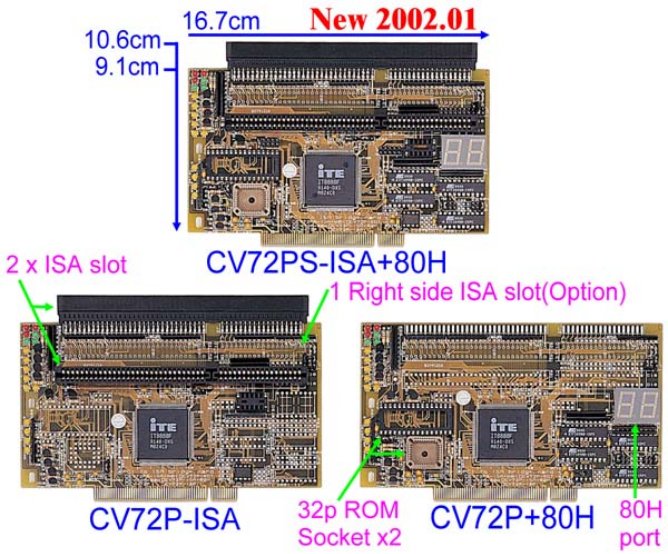


Architettura basata su bus PCI

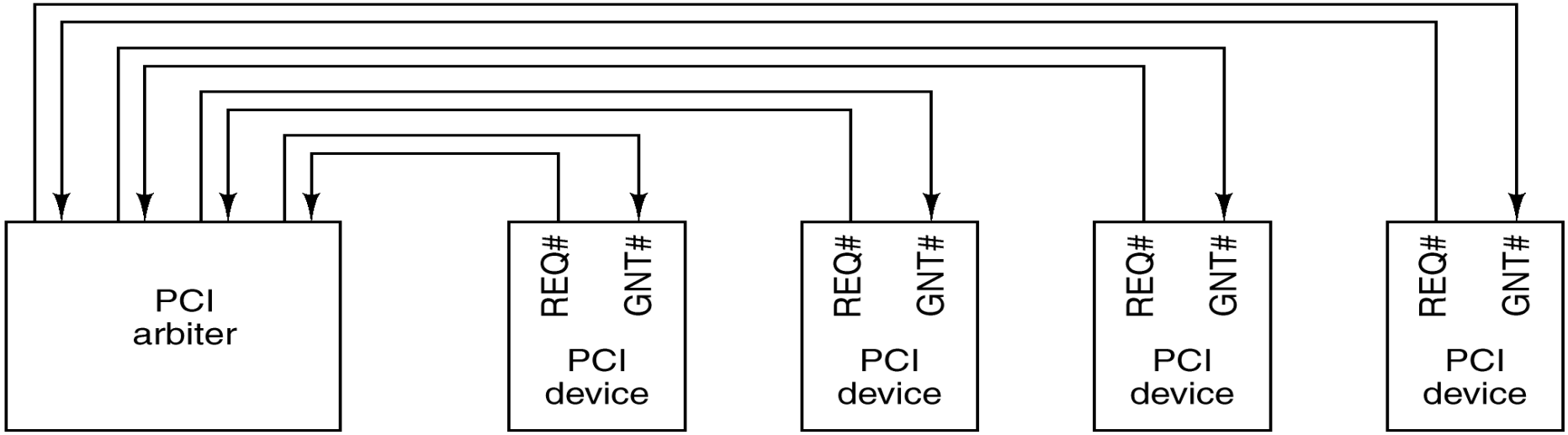


Il Bus PCI: specifiche

- **PCI** (**P**eripheral **C**omponent **I**nterconnect)
- Introdotto da Intel per applicazioni video
- Standard *non proprietario*, adottato da molti
- Versione base a 32 bit, 33 MHz: 133 MB/sec
- Estensione a 64 bit e 66 MHz: 528 MB/sec
- Memory Bus separati (più veloci)
- Connessione tramite chip **PCI bridge**
- Varie opzioni di tensione (5 V e 3.3 V)
- Schede con 120 e 120+64 contatti
- **Bridge ISA** (include doppio controller IDE)
- Controllori addizionali SCSI e USB
- Bus sincrono, transazioni tra **initiator** e **target**
- Linee indirizzo e dati condivise



Il Bus PCI: arbitraggio



- Arbitraggio centralizzato (nel Bridge)
- Ogni PCI device ha due linee dedicate
- Il dispositivo fa la richiesta tramite REQ#
- Il grant viene concesso tramite GNT#
- Diversi algoritmi di arbitraggio:
  - Round Robin
  - Priorità
  - Altro
- Transazioni su più cicli separate da cicli di idle

Il Bus PCI: Segnali Obbligatori

Signal	Lines	Master	Slave	Description
CLK	1			Clock (33 MHz or 66 MHz)
AD	32	×	×	Multiplexed address and data lines
PAR	1	×		Address or data parity bit
C/BE	4	×		Bus command/bit map for bytes enabled
FRAME#	1	×		Indicates that AD and C/BE are asserted
IRDY#	1	×		Read: master will accept; write: data present
IDSEL	1	×		Select configuration space instead of memory
DEVSEL#	1		×	Slave has decoded its address and is listening
TRDY#	1		×	Read: data present; write: slave will accept
STOP#	1		×	Slave wants to stop transaction immediately
PERR#	1			Data parity error detected by receiver
SERR#	1			Address parity error or system error detected
REQ#	1			Bus arbitration: request for bus ownership
GNT#	1			Bus arbitration: grant of bus ownership
RST#	1			Reset the system and all devices

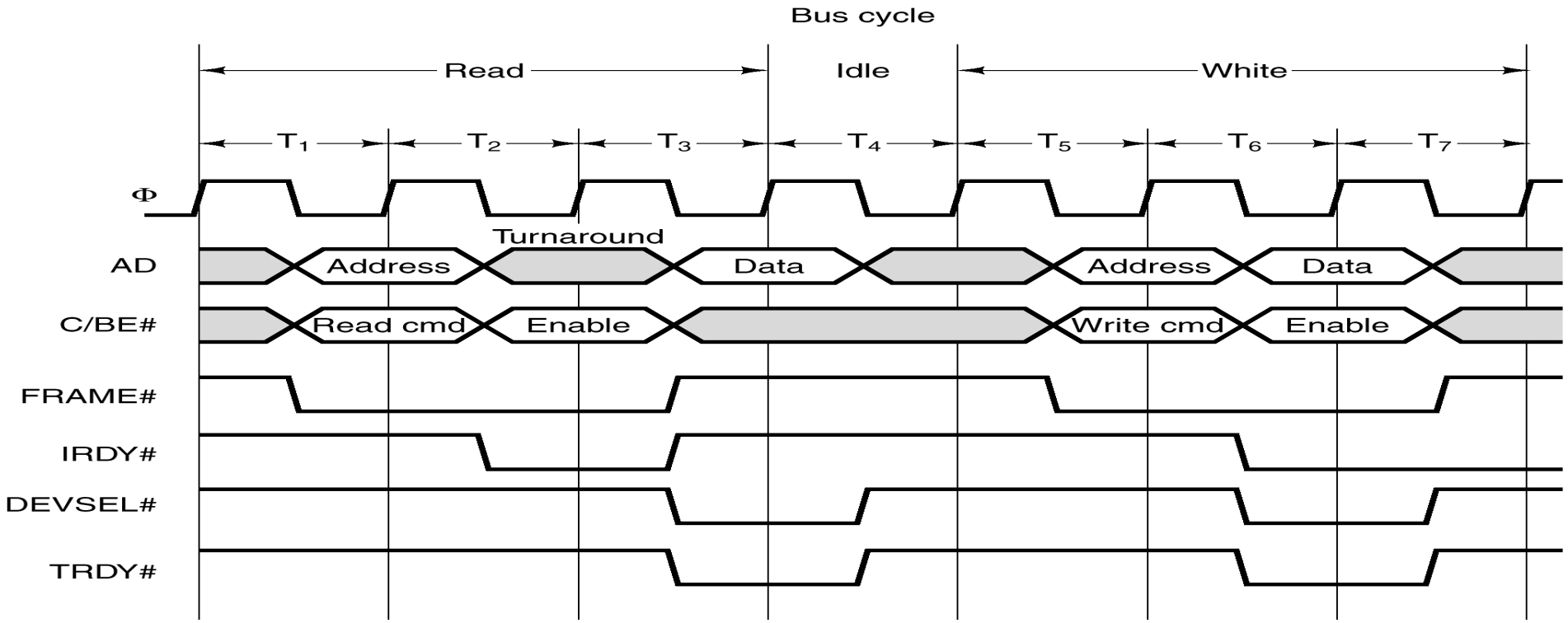
Il Bus PCI: Segnali Opzionali

Sign	Lines	Master	Slave	Description
REQ64#	1	×		Request to run a 64-bit transaction
ACK64#	1		×	Permission is granted for a 64-bit transaction
AD	32	×		Additional 32 bits of address or data
PAR64	1	×		Parity for the extra 32 address/data bits
C/BE#	4	×		Additional 4 bits for byte enables
LOCK	1	×		Lock the bus to allow multiple transactions
SBO#	1			Hit on a remote cache (for a multiprocessor)
SDONE	1			Snooping done (for a multiprocessor)
INTx	4			Request an interrupt
JTAG	5			IEEE 1149.1 JTAG test signals
M66EN	1			Wired to power or ground (66 MHz or 33 MHz)

Bus PCI: Segnali

- Bus a 120 o 180 linee
- Oltre ai segnali sono distribuiti anche alimentazioni e masse
- 32(+32) linee AD condivise tra dati e indirizzo con 1(+1) bit di parità PAR
- C/BE# (in cicli diversi) invia comandi e specifica quanti e quali byte sono validi
- FRAME# e IRDY usati dal master
- DEVSEL# e TRDY# usati dallo slave
- IDSEL indirizza lo spazio di configurazione di un dispositivo invece che la memoria (usato per gestire il *Plug and Play*)
- STOP# e PERR# segnalano errori
- RST# induce un reset nella CPU e in tutti i device sul bus
- SBO e SDONE segnali di cache snooping

Bus PCI: Transazioni



- In T1 il master invia l'indirizzo su AD e il comando su C/BE#
- Poi asserisce FRAME# e poi IRDY#
- In T2 C/BE# specifica quali byte leggere
- In T3 lo slave asserisce DEVSEL# e quando i dati sono su AD asserisce TRDY#
- Tra due transazioni c'è un ciclo di idle
- La transazione di scrittura è più compatta

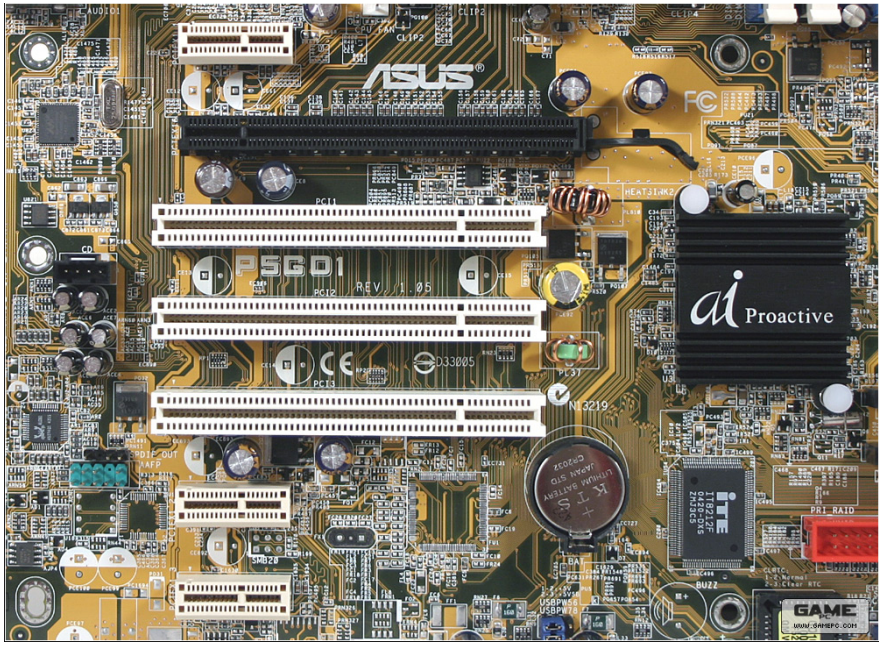
PCI express

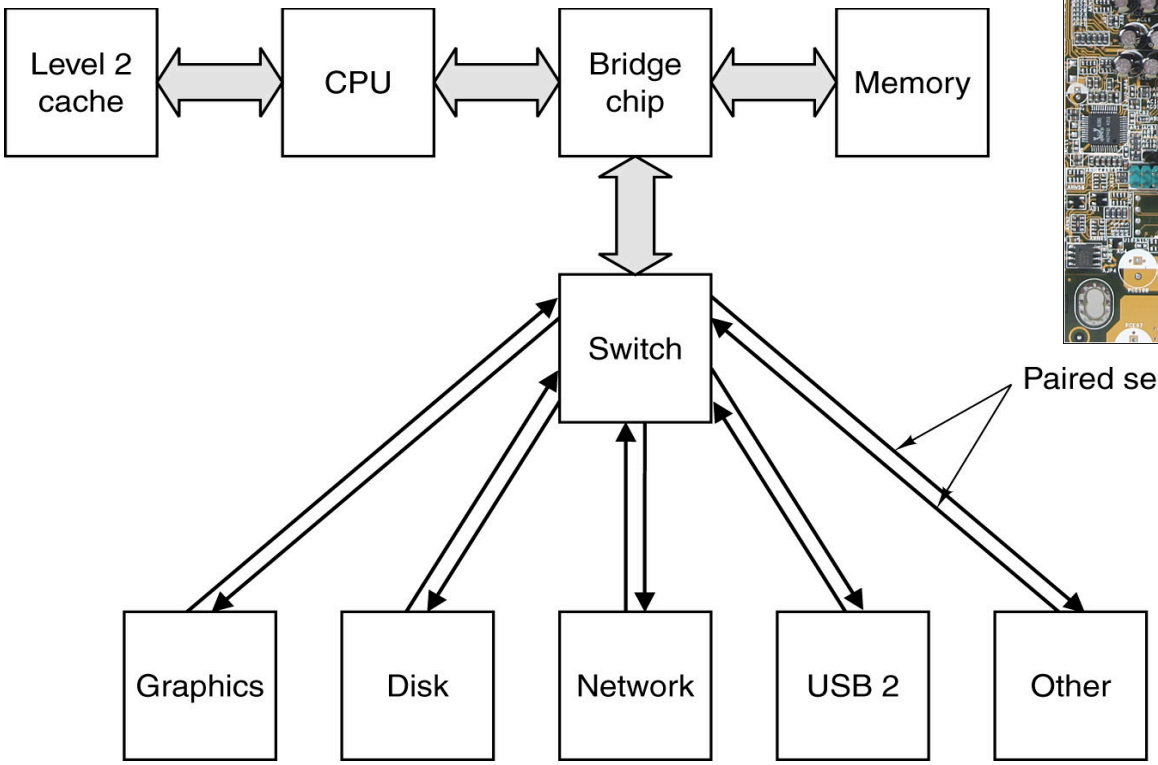
Linea di tendenza nei bus:

- bypassare il bus PCI nel caso di periferiche veloci
- usare slot più piccole

Soluzione: PCI express

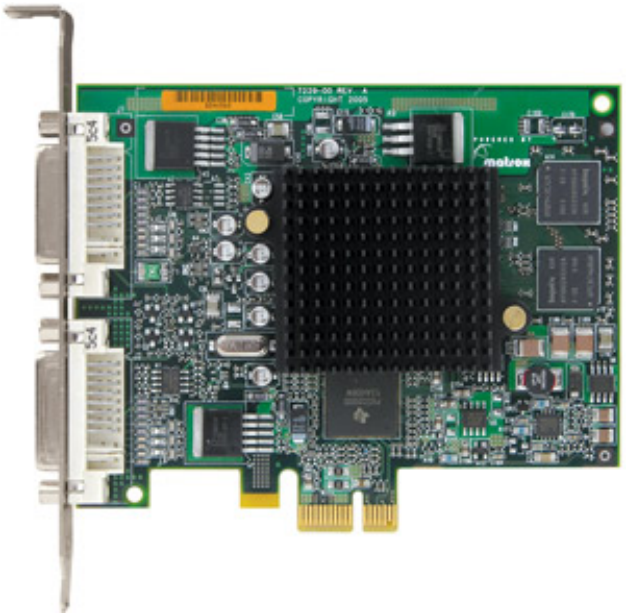
- Connessione punto-a-punto





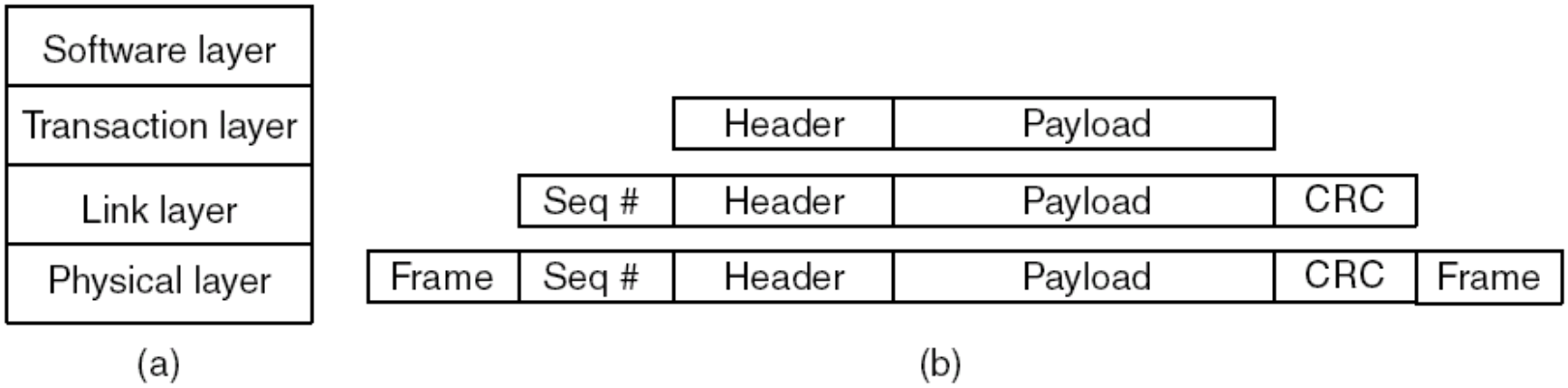
Caratteristiche PCI express

- Trasmissione seriale
- Trasferimenti come in una rete di computer:
  - dati in pacchetti (header + payload + CRC)
  - maggiore lunghezza dei cavi
  - plug-and-play
  - connettori più piccoli
- Banda: attualmente 16GB/sec (ver. 3.0)
- Controllo del flusso sulla base delle dimensioni dei buffer
- 4 spazi di indirizzamento:
  - Memoria
  - I/O
  - Configurazione
  - Messaggi



PCI Express Protocol Stack

- Trasmissione basata su protocollo multi-layer lungo coppie di corsie (lane)
- Codifica: 8b/10b
- Un meccanismo di acknowledgment garantisce maggiore affidabilità
- Il software layer garantisce:
  - la gestione dei pacchetti
  - la compatibilità con il passato

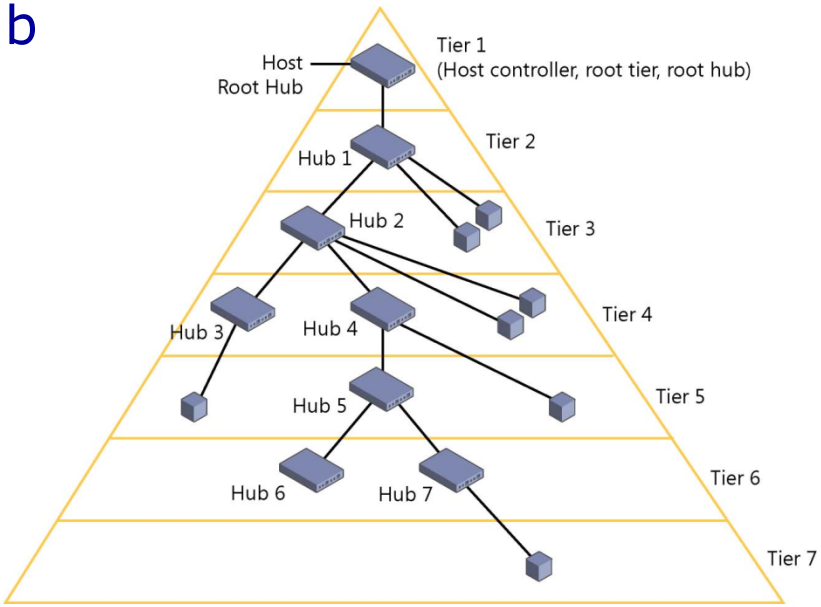


Bus USB (Universal Serial Bus)

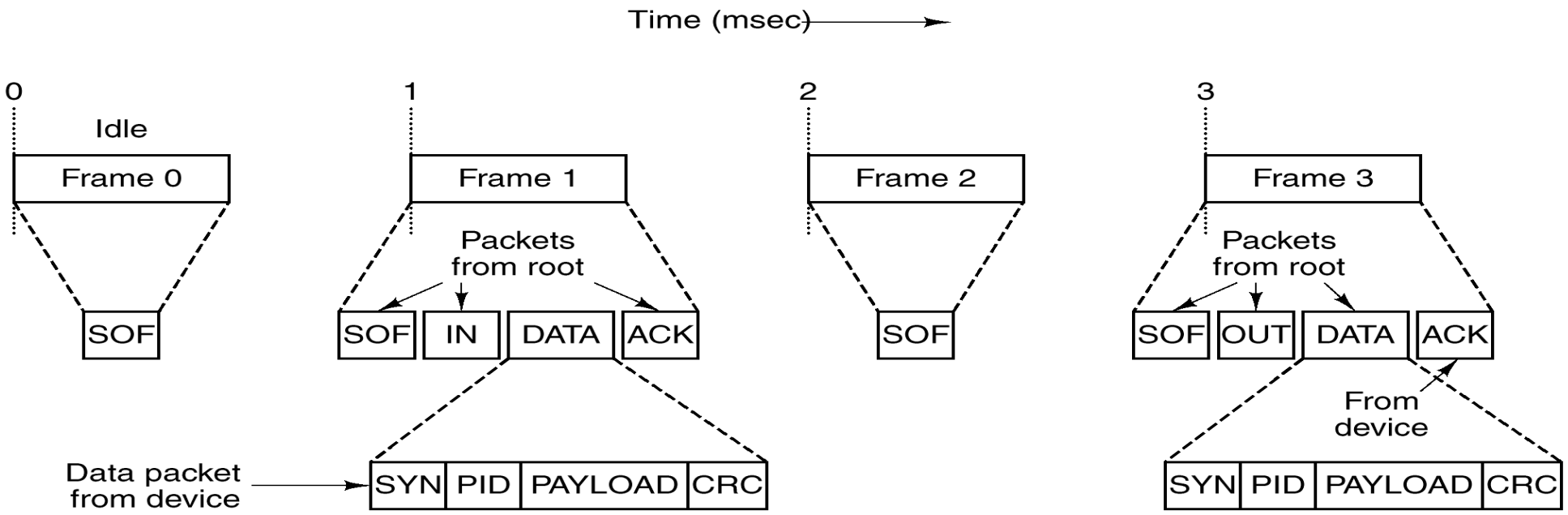
- Bus economico concordato da varie aziende per la gestione di dispositivi di I/O a bassa velocità (~ 1995)
- Obiettivi:
  1) Evitare *switch*, *jumpers*
  2) Installazione di tipo esterno
  3) Cavo di connessione unificato
  4) Alimentazione fornita dal cavo
  5) Fino a 127 dispositivi collegabili
  6) Supporto di dispositivi real-time
  7) Installazione a PC acceso
  8) Reboot non necessario
  9) Bus e dispositivi economici
- Tutti gli obiettivi sono di fatto rispettati

USB: Specifiche Fondamentali

- Banda complessiva
  - USB 1.x: 1.5 – 12 Mb/sec
  - USB 2.0: 480 Mb/sec
  - USB 3.0: 5 Gb/sec
- *Root hub* di connessione al bus PCI
- Connessione di dispositivi e di altri hub
- Struttura complessiva ad albero
- Connettori ai capi del cavo diversi
- Cavo a 4 fili: +5V, GND, 2 di segnale
- Alla connessione di un dispositivo:
  - Interrupt: intervento del SO
  - Richiesta di banda
  - Assegnazione di indirizzo
- Indirizzo 0 usato per inizializzazione
- Logicamente: connessione dedicata tra root hub e ciascun device
- Competitor: FireWire IEEE 1394 serial bus (400 Mb/sec)

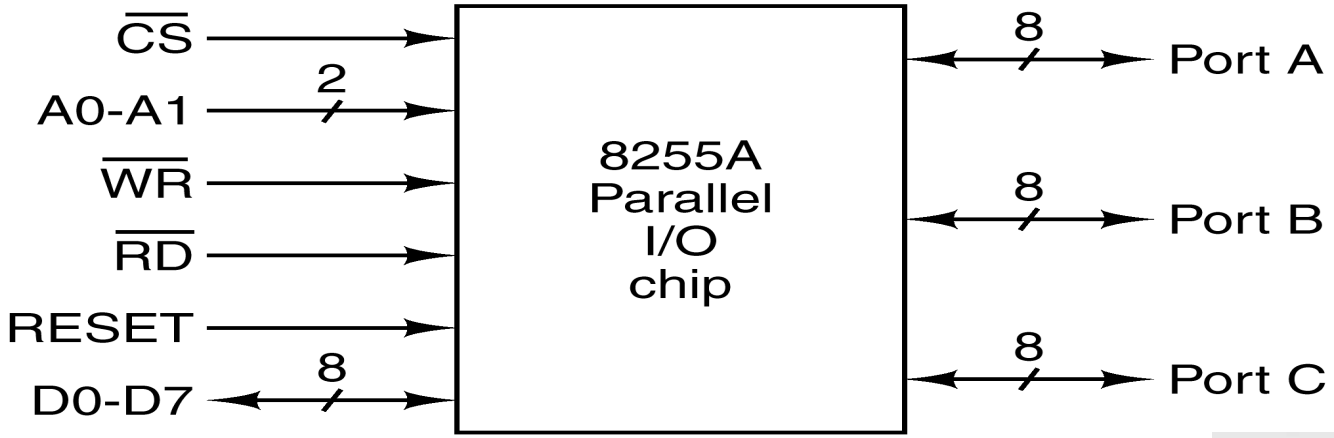


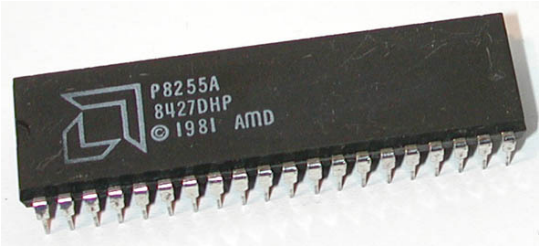
USB: Struttura dei Frame

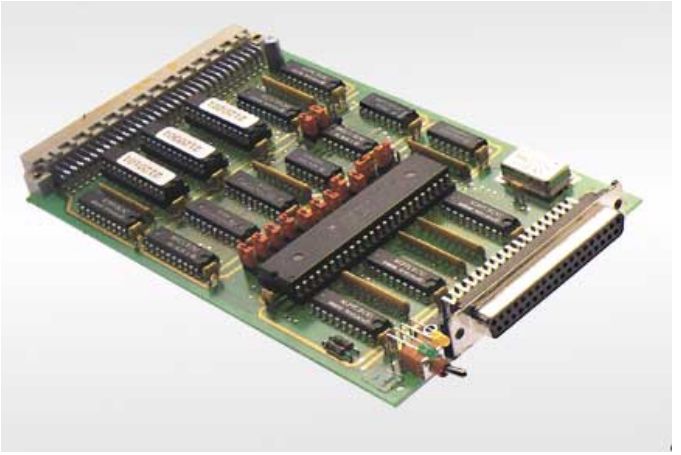


- Frame emessi ogni 1.00±0.05 msec
- Idle frame se non c'è comunicazione
- Contenuto del frame:
  - SOF: Start of Frame
  - IN / OUT: richiesta in lettura/scrittura
  - DATA: payload fino a 64 byte più controllo e codice di errore
  - ACK / NACK: acknowledge o errore
- Polling usato invece delle interruzioni

Chip di I/O: UART, USART e PIO

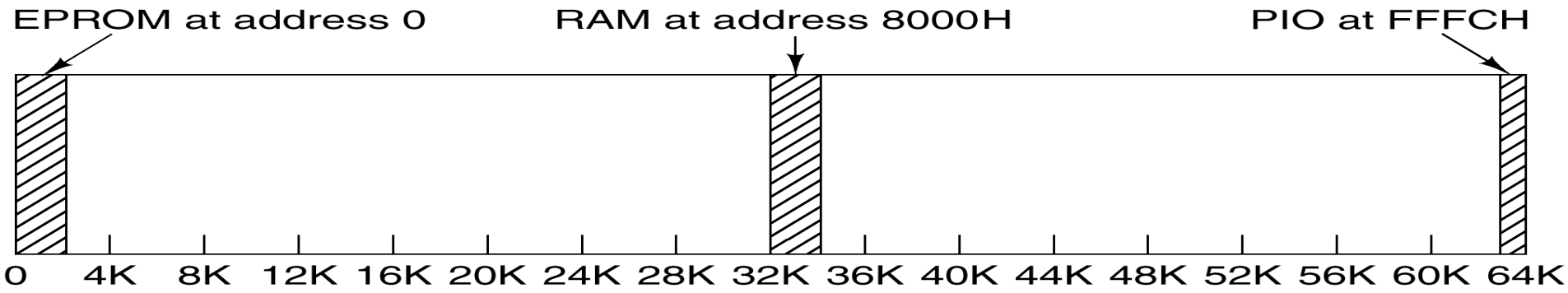






- UART (Univ. Async. Rec. Transm.)
- USART (… Sync. Async. ……)
  - Usati in interfacce parallelo/seriale
- PIO (Parallel Input/Output)
  - Configurabile dalla CPU
  - 3 porte indipendenti da 8 bit con latch
  - La CPU legge e scrive direttamente nelle porte
  - Possibile gestire anche semplici protocolli di handshaking CPU-device

Decodifica degli Indirizzi

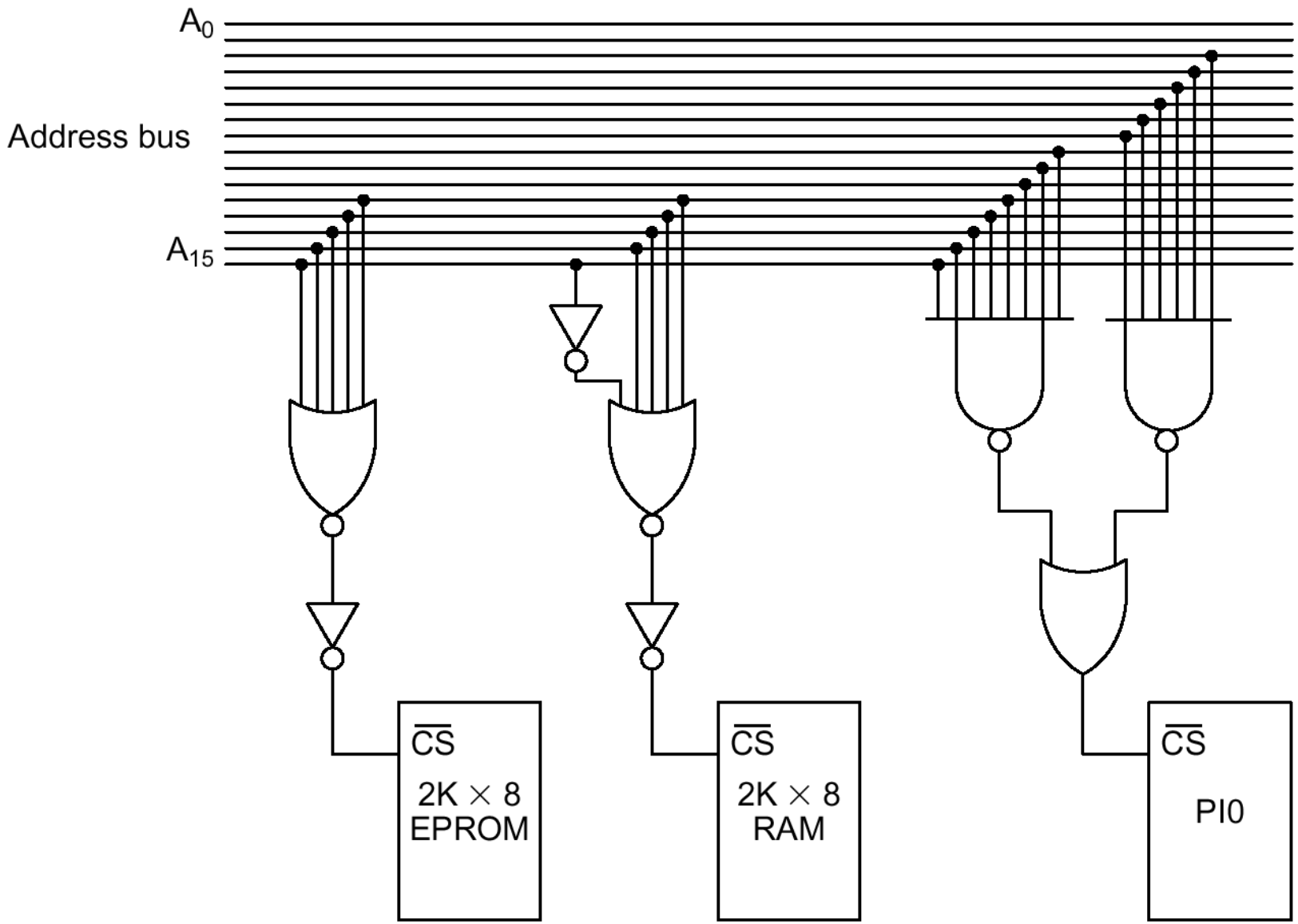


- Spazio separato di I/O, oppure unico spazio (Memory-Mapped I/O)
- Alcuni indirizzi riservati alle porte di I/O
- Problema: ricavare i segnali di chip select

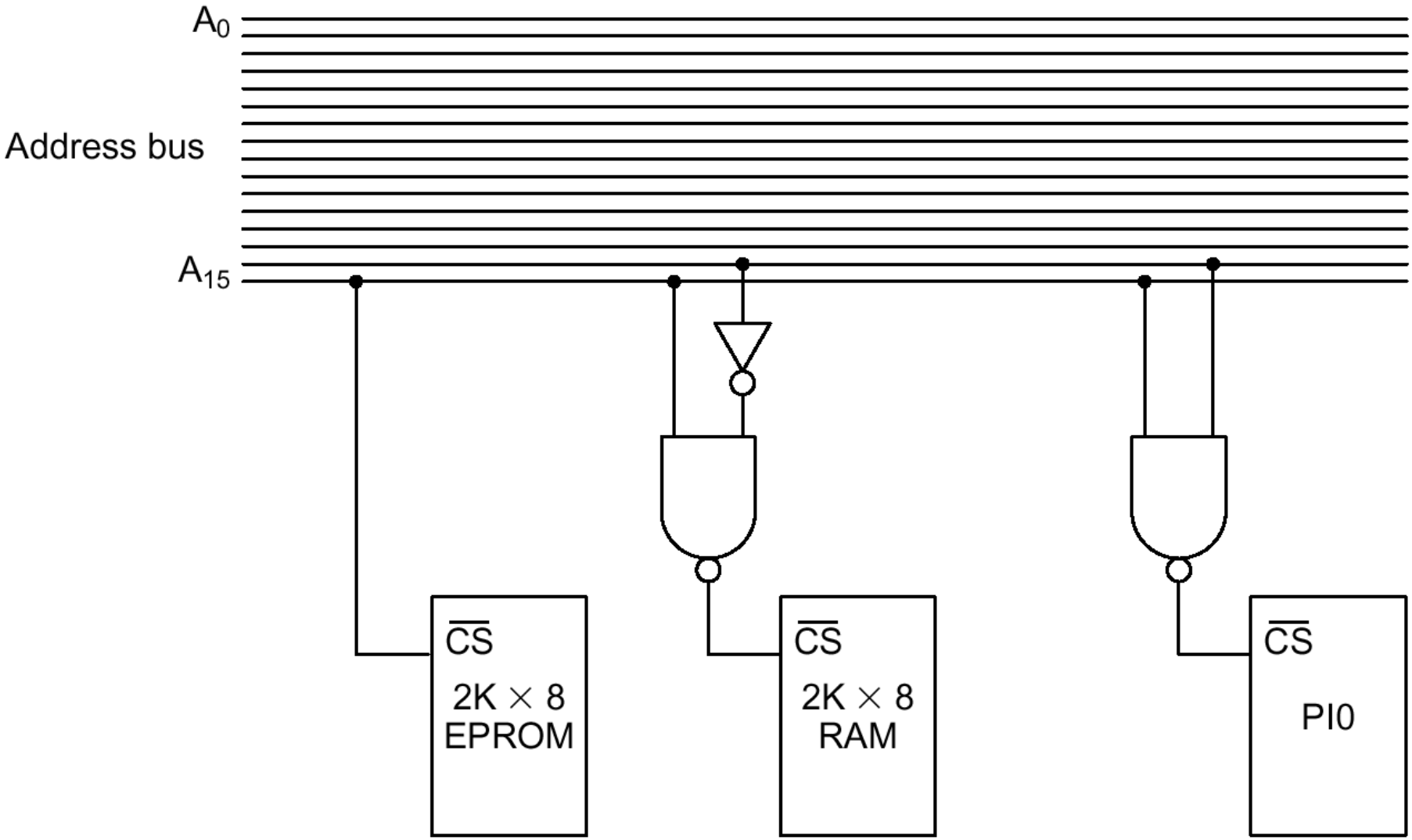
ES

- EPROM: 0-2K
  0000 0000 0000 0000
  0000 0111 1111 1111
- RAM: 32K-34K
  1000 0000 0000 0000
  1000 0111 1111 1111
- PIO: FFFC-FFFF
  1111 1111 1111 1100
  1111 1111 1111 1111

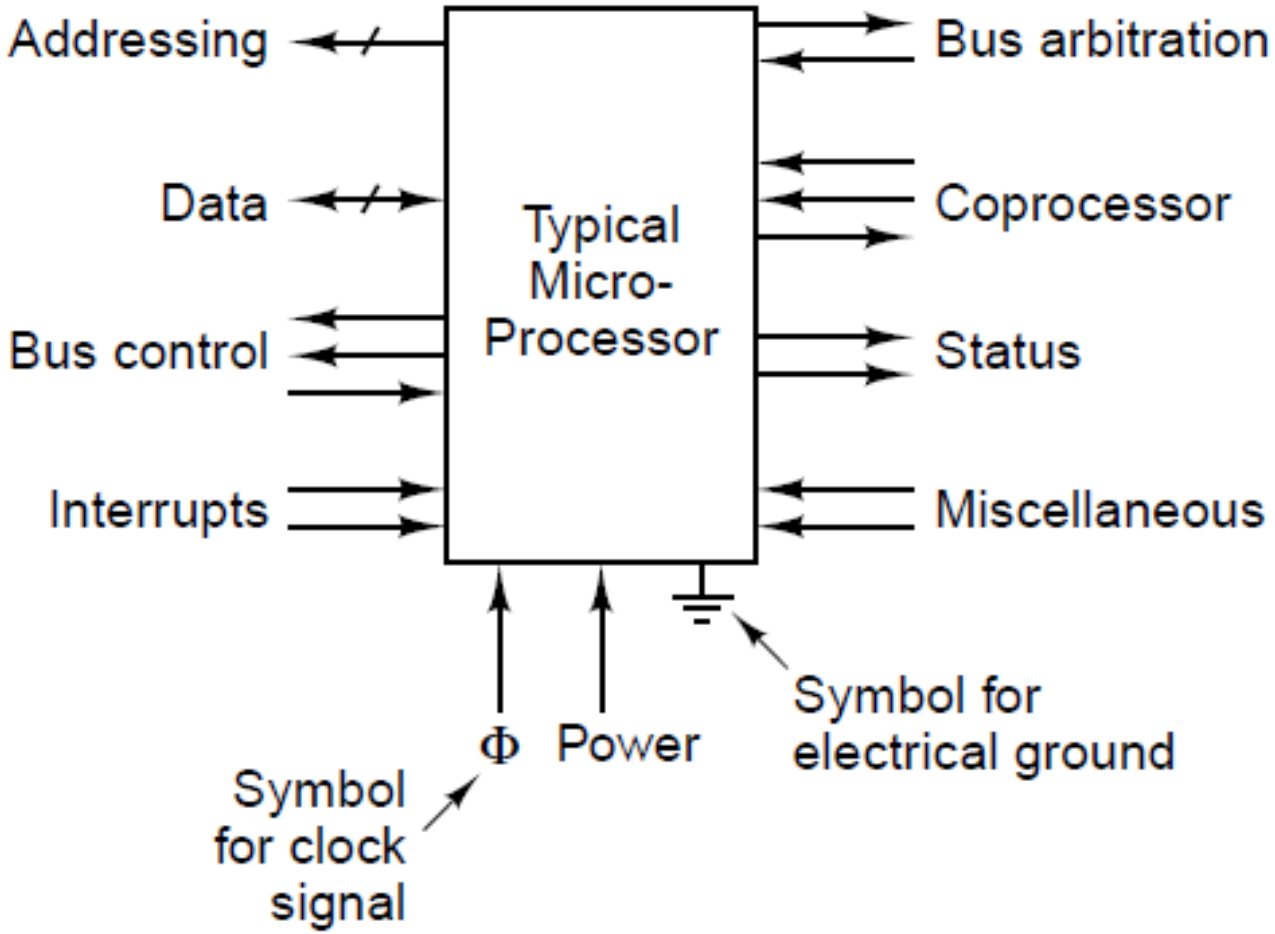
Decodifica Completa



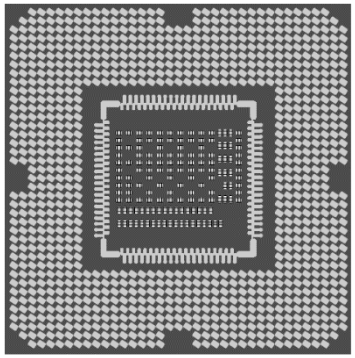
Decodifica Parziale



Pinout Logico di un Micro-Processore

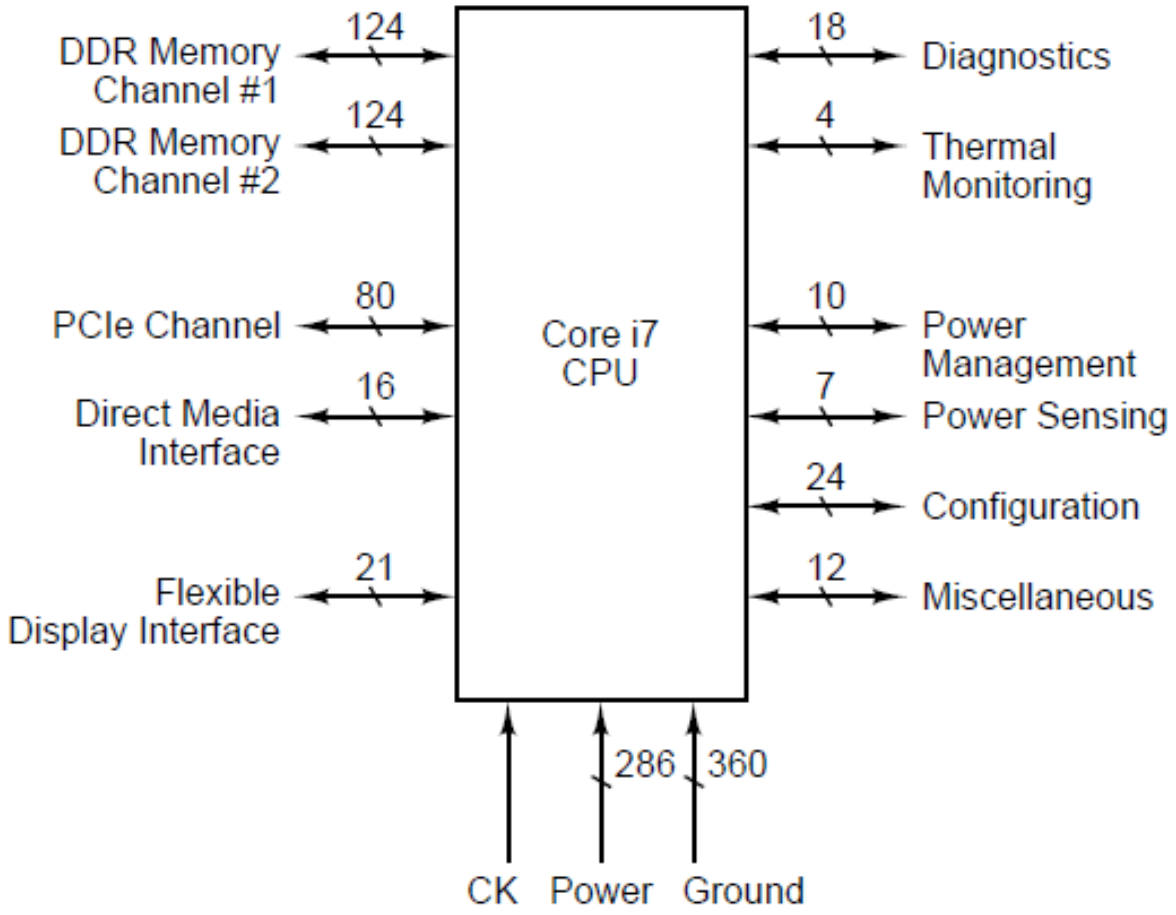


Intel Core i7



- Nahalem (2008): 731 milioni di transistor, 45 nm, 3.2 Ghz
- Sandy Bridge (2011): 1,16 miliardi di transistor, 32nm, 3.5 Ghz
- Architettura a 64 bit compatibile con i predecessori
- Aritmetica Floating-point IEEE 754
- Architettura multicore (2-6)
- Hyperthreaded, superscalare (fattore 4), pipelined
- Bus di memoria sincrono a 64 bit + Bus PCIe
- QPI (Quick Path Interconnect): comunicazione con altri processori
- Cache 1° livello 32KB dati + 32KB istruzioni
- Cache 2° livello 256 KB per core (snooping)
- Cache 3° livello condivisa da 4 a 15 MB
- Scheda con 1155 pin (diversa dai predecessori)
- Consuma da 17 a 150W (stati differenti per ridurre il consumo)

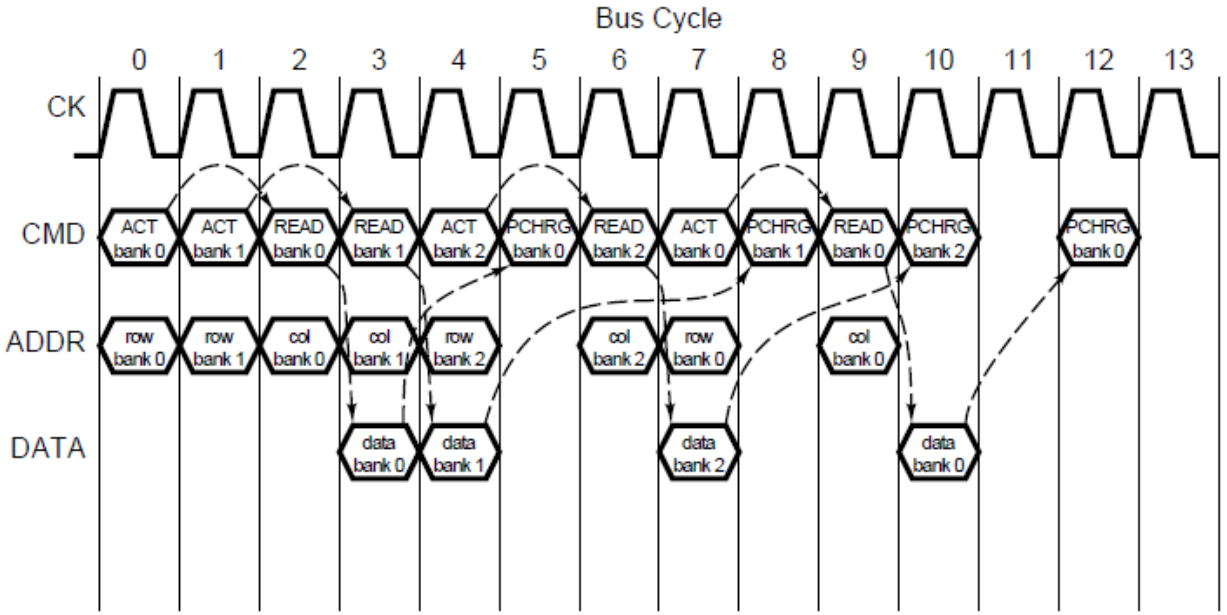
Intel Core i7: Pinout Logico



Intel Core i7: Pinout Logico (2)

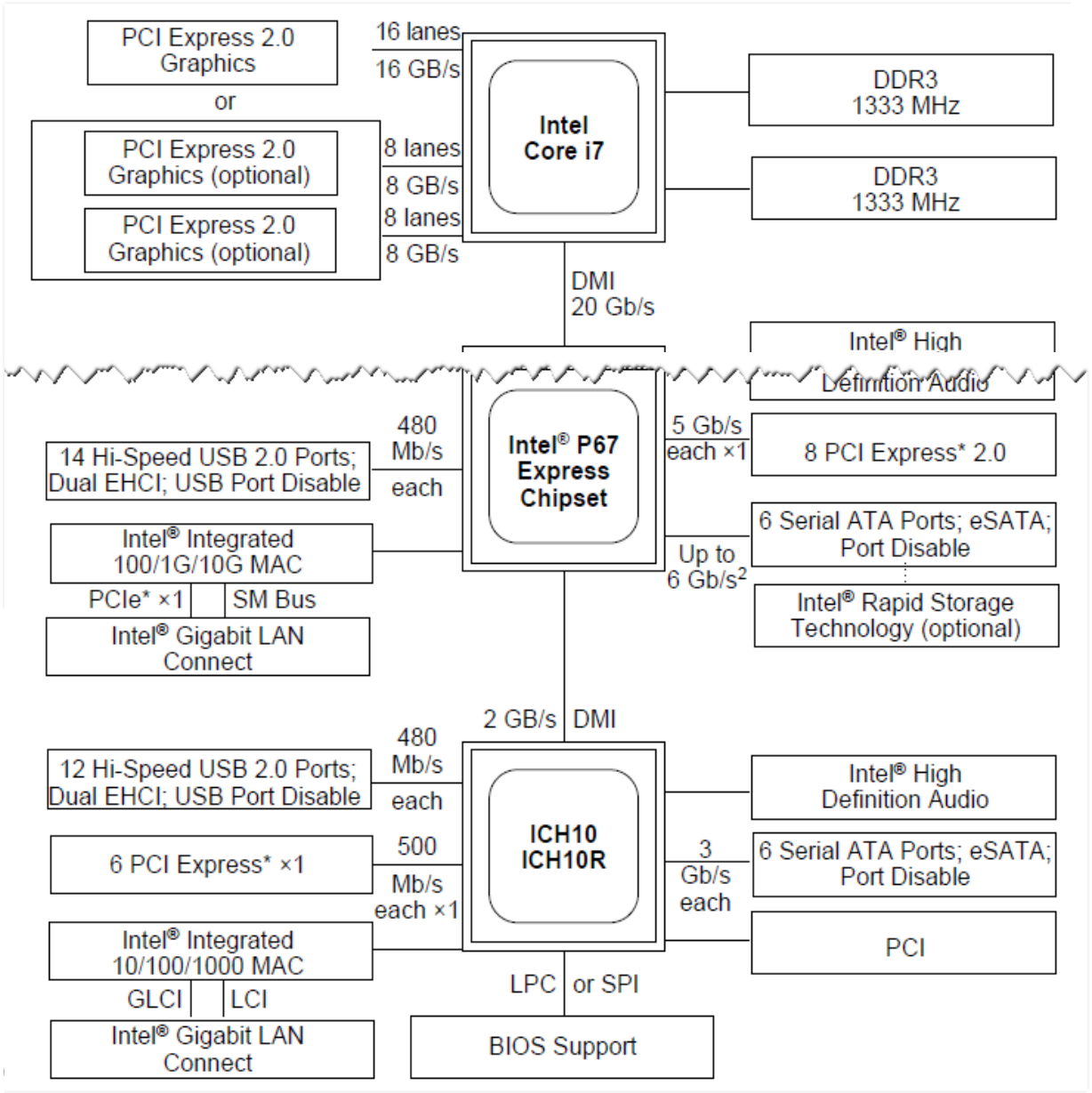
- 1155 piedini
  - 447 per i segnali (alcuni duplicati, 131 in tutto)
  - 286 connessioni di alimentazione
  - 360 connessioni di massa
  - 62 per “uso futuro”
- Due gruppi indipendenti per l’interfaccia con una DRAM
  - 64bit, 666Mhz, 1,333 MTPS, 20 GB/sec complessivi
- Un gruppo per l’interfaccia con linee PCIe
  - 16 linee (lane), 16GB/sec complessivi
- Un gruppo per la comunicazione con il chipset (DMI)
  - 4 linee, 2.5GB/sec complessivi
  - P67: SATA, USB, Audio, PCIe, Flash;
  - ICH10: PCI, 8259A, clock, timer, controllori DMA
- Gestione delle interruzioni sia come l’8088 che con APIC (Advanced Programmable Interrupt Controller)
- Gestione della tensione: (possibili diversi valori di Voltaggio)
- Thermal monitoring: sensori di calore per il "thermal throttling"
- 11 linee di diagnosi secondo lo standard IEEE 1149.1 JTAG

Intel Core i7: Memory Bus



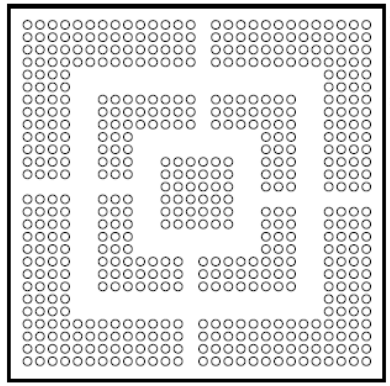
- Bus gestito con pipelining: è possibile sovrapporre 4 transazioni
- Ogni interfaccia DRAM ha 3 gruppi di linee
- Fasi di una transazione (usano gruppi di linee indipendenti):
  - Attivazione e invio indirizzi
  - Comando di Read/Write di parole contigue della memoria (bank)
    - Richiede due passi: comando e trasferimento dati
  - Chiusura e preparazione per la prossima transazione
- Funziona solo con memorie sincrone

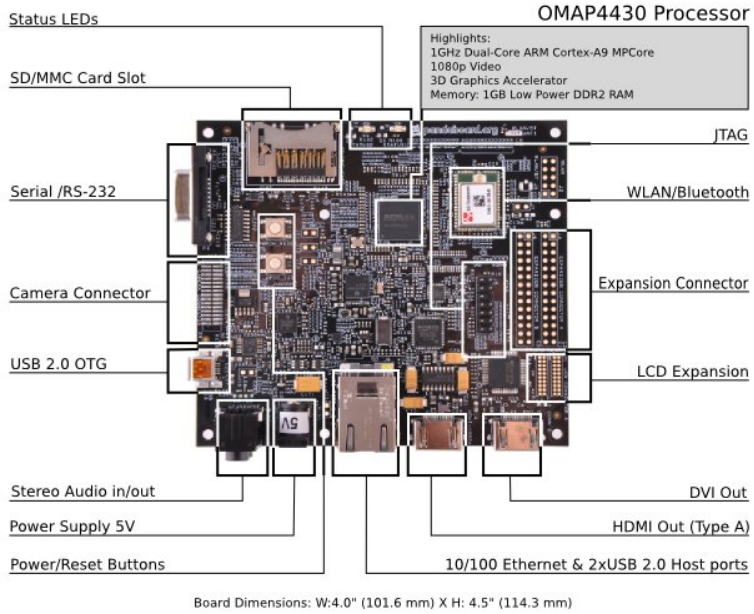
Struttura di un sistema moderno basato su i7

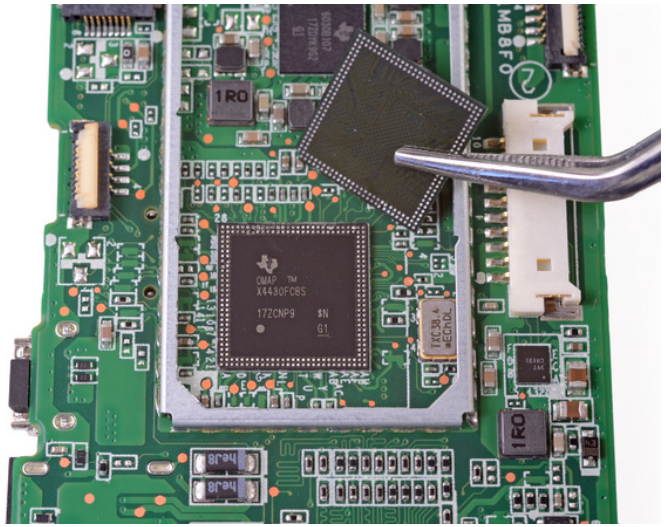


OMAP4430 della Texas Instruments

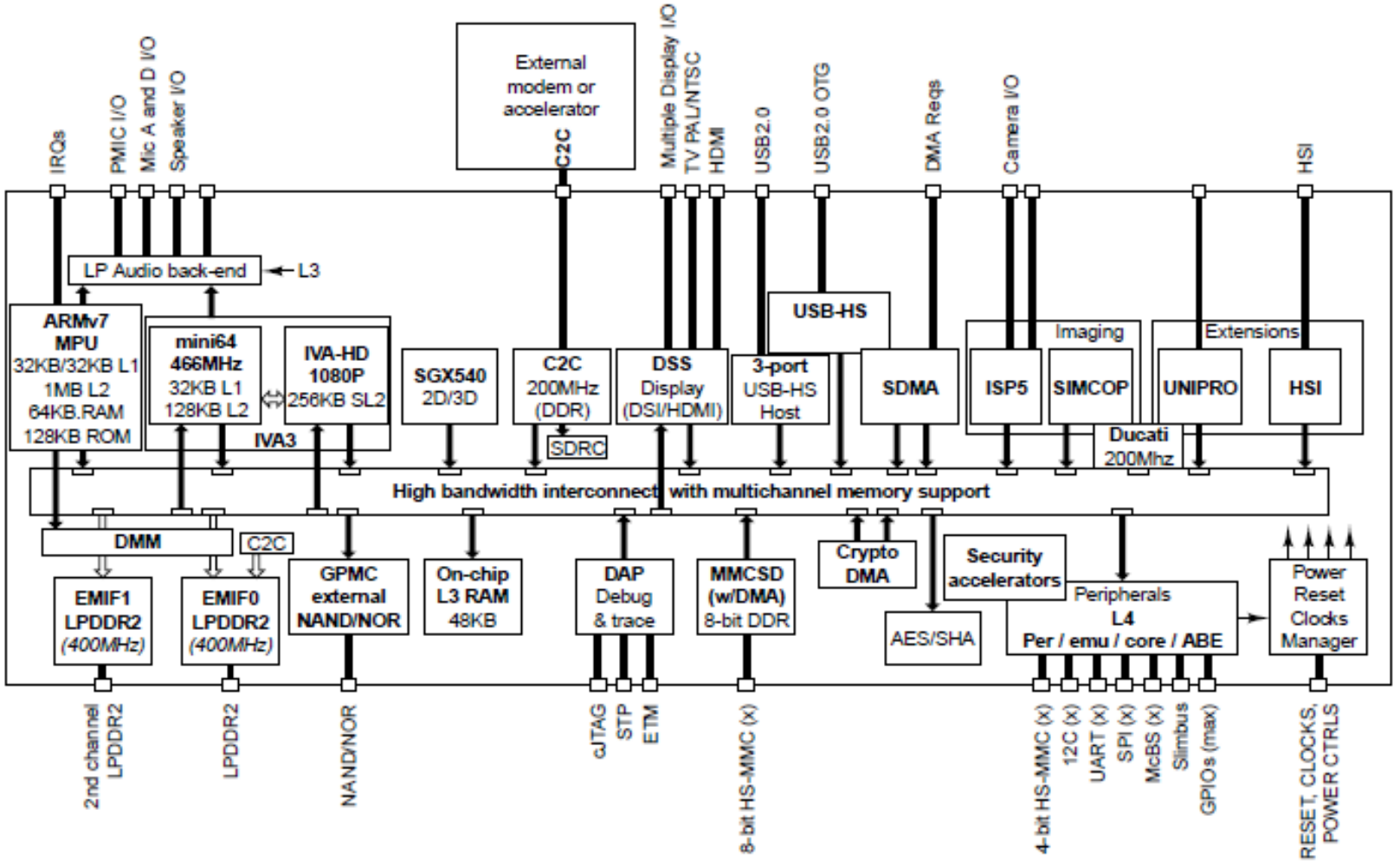
- SoC basato su ISA ARM
- Target: sistemi mobile o embedded
- Equipaggiamento:
  - 2 CPU ARM RISC Cortex-A9 a 1Ghz, 45nm
  - 1 GPU POWERV SGX540 (rendering 3D)
  - 1 ISP (manipolazione immagini)
  - 1 VPU IVA3 (video enc/dec)
- Interfacce I/O:
  - Touchscreen, keypad
  - DRAM, Flash
  - USB, HDMI
- Basso consumo di potenza
  - 660mW/100µW
  - dynamic voltage scaling
  - power gating
- Superscalare (2 istr+branch perd.+OoO)
- 2 L1: 32KB+32KB, 1 L2: 1MB
- Interfaccia DRAM (LPDDR2)
- Scheda a 547 pin





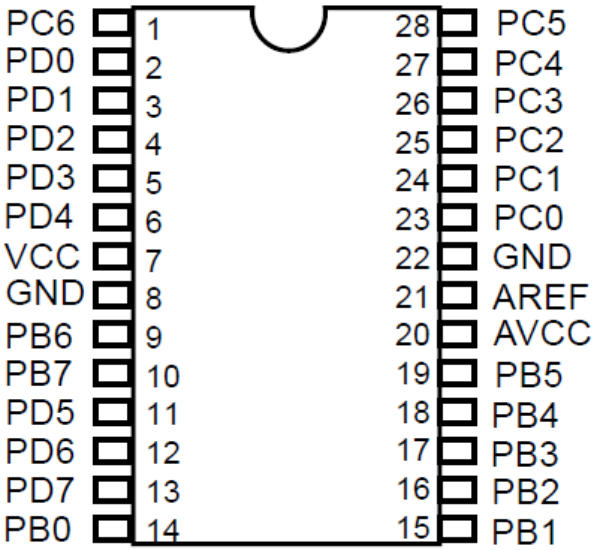


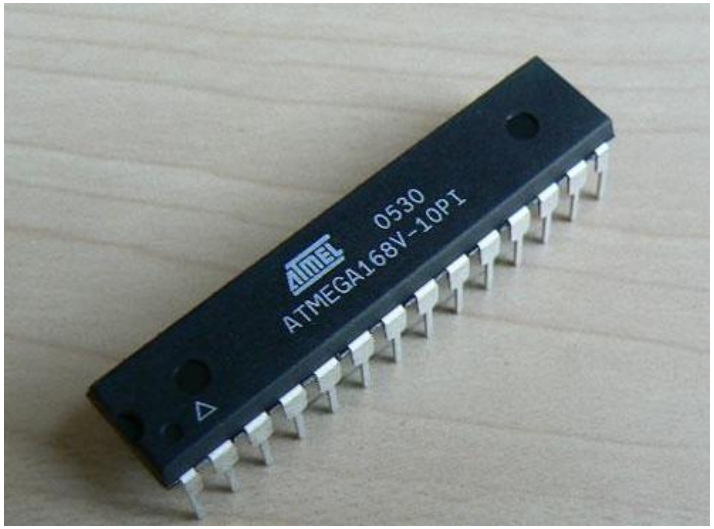
Architettura OMAP4430



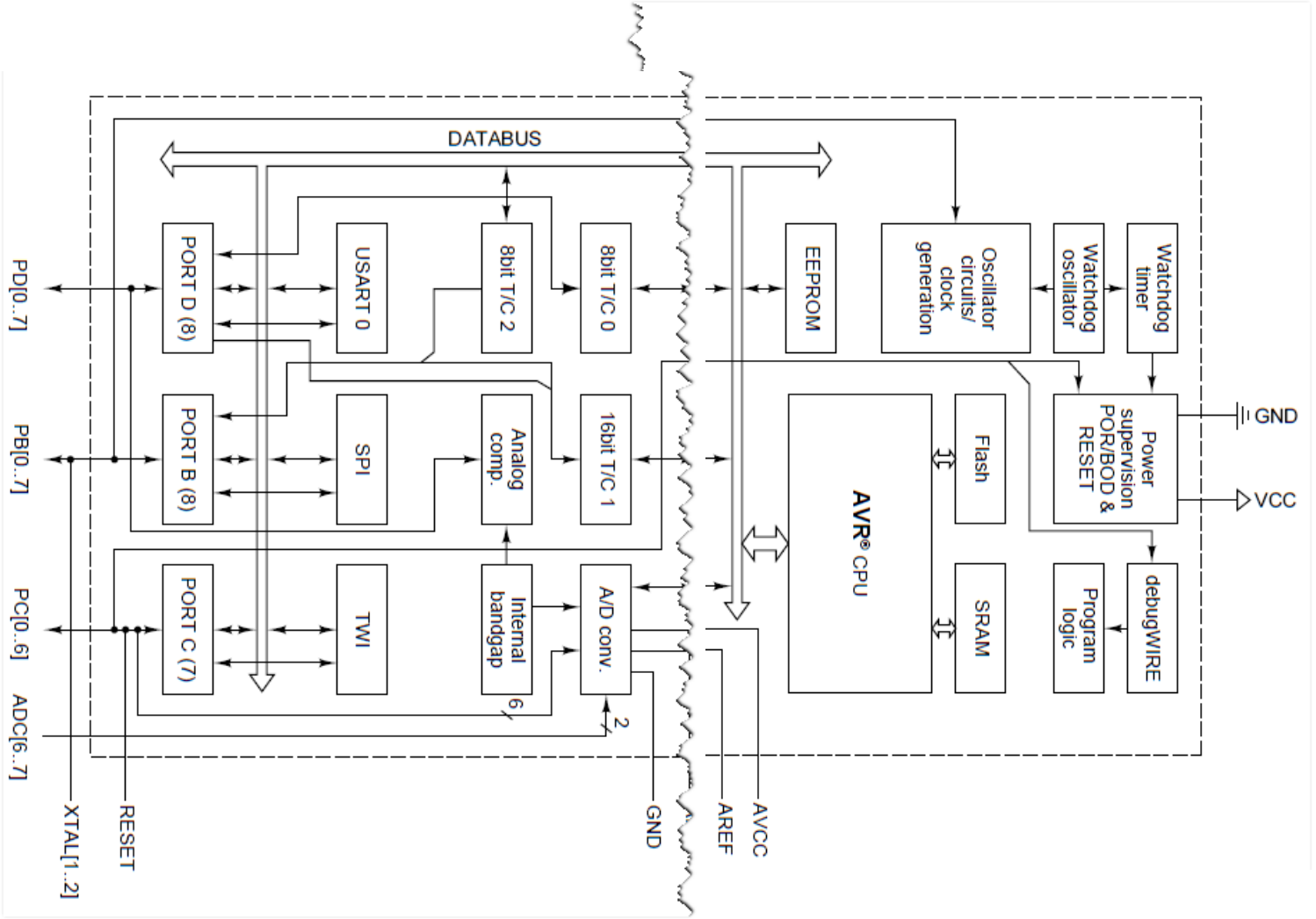
Atmel ATmega168

- Microcontrollore per applicazioni embedded (~1$)
- CPU a 8 bit basata su ISA AVR
- Scheda a 28 pin
  - 23 porte di I/O
    - 8 per porte B e D
    - 7 per porta C (analogica)
  - 1 Vcc+2GND
  - 2 per configurare circuiti analogici
- Memorie incorporate
  - 16KB Flash
  - 1KB EEPROM
  - 1KB SRAM
- No RAM esterna
- Clock real time
- Interfaccia seriale





Architettura ATmega168



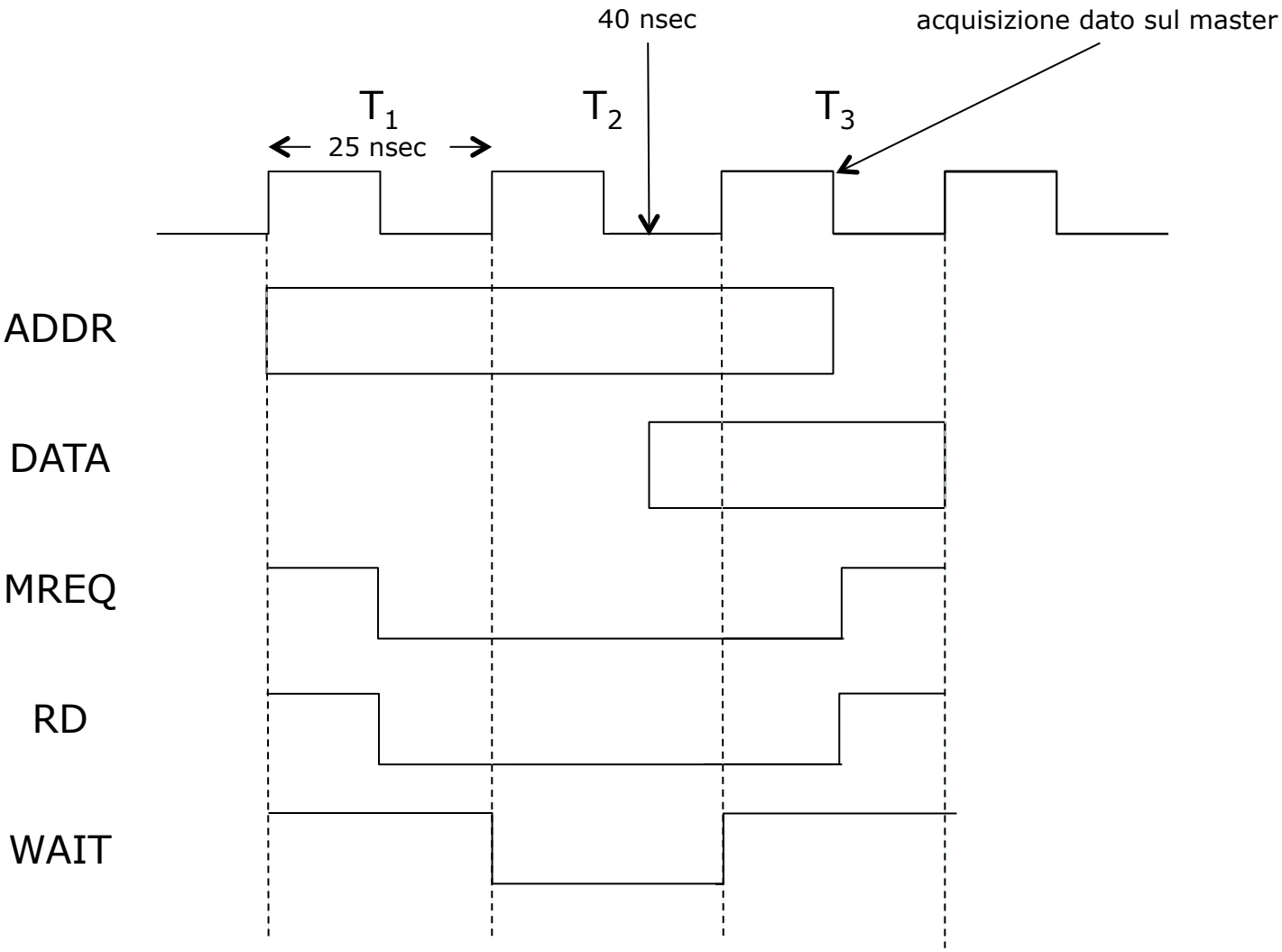
Esercizio su BUS

Con riferimento al funzionamento dei bus di un calcolatore:

- tracciare e illustrare il diagramma di temporizzazione di un bus sincrono a 40 Mhz *con linee separate per dati e indirizzi* e segnali di $\overline{\text{MREQ}}$, $\overline{\text{RD}}$ e $\overline{\text{WAIT}}$, per una lettura da una memoria con un tempo di risposta di 40 nsec *dal momento in cui gli indirizzi sono disponibili*

Si assuma di lavorare in condizioni ideali (nessun ritardo nell'asserimento dei segnali)

Soluzione esercizio su BUS



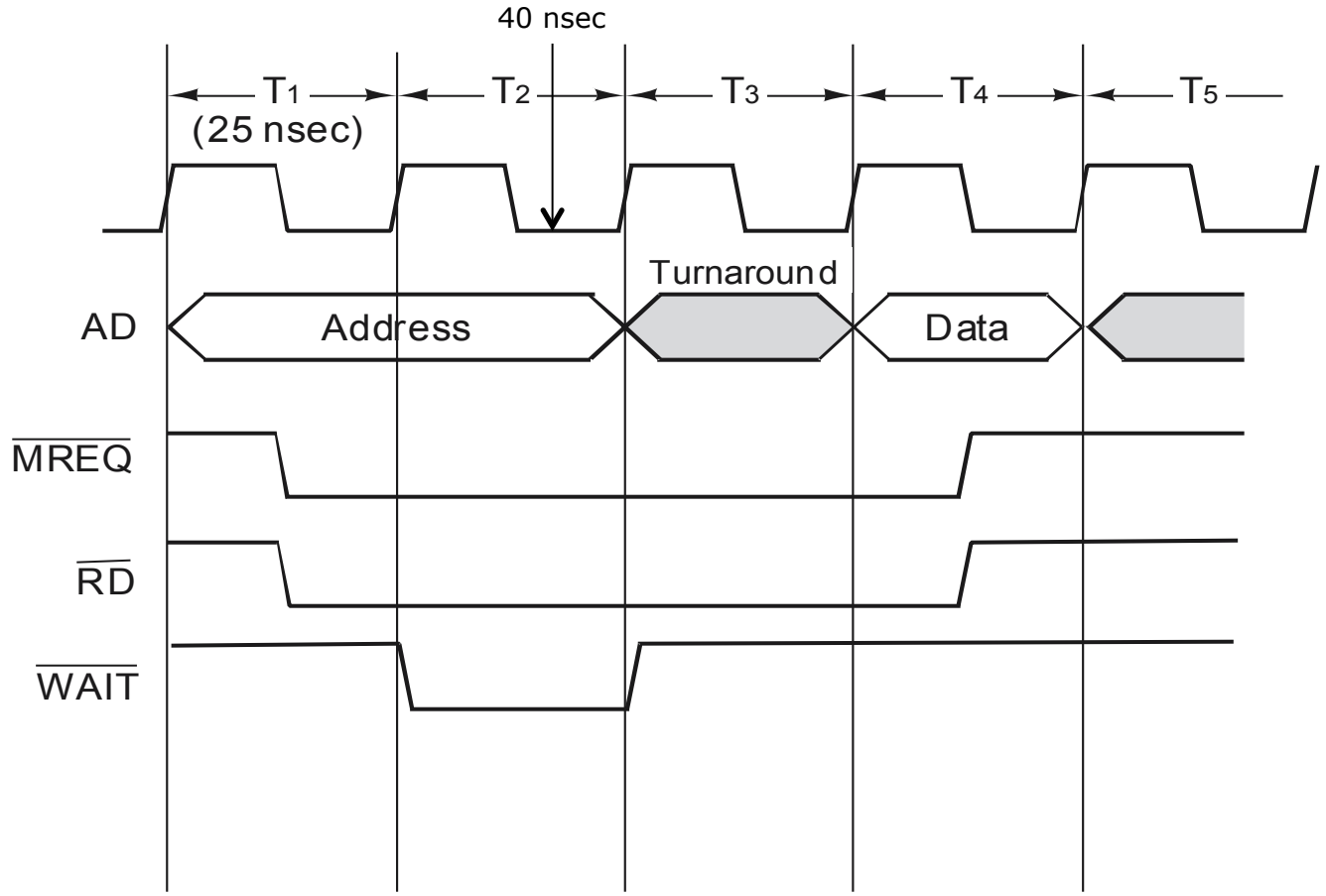
Esercizio su BUS

Con riferimento al funzionamento dei bus di un calcolatore:

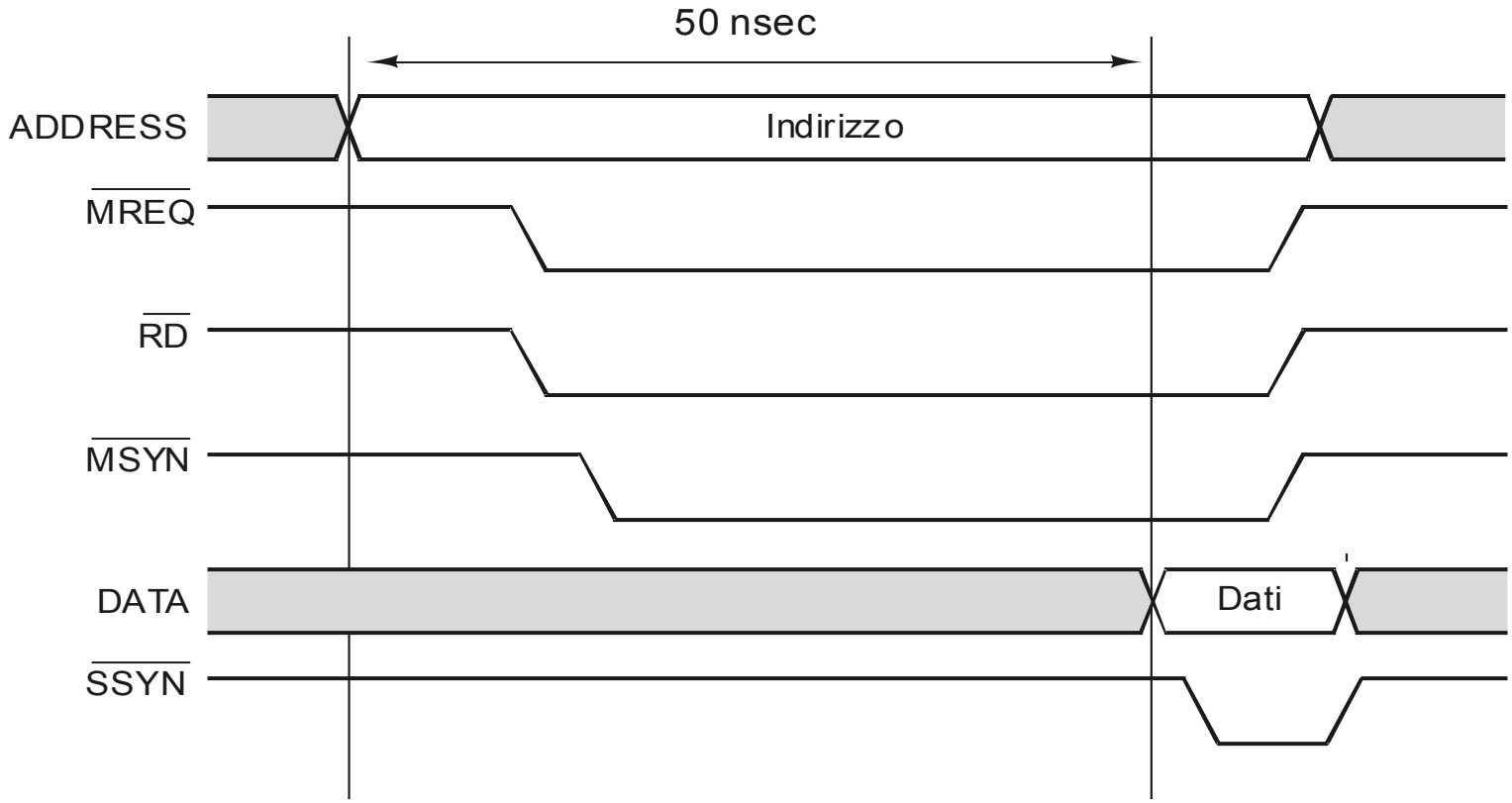
- tracciare e illustrare il diagramma di temporizzazione di un bus sincrono a 40 Mhz *con linee condivise per dati e indirizzi* e segnali di $\overline{\text{MREQ}}$, $\overline{\text{RD}}$ e $\overline{\text{WAIT}}$, per una lettura da una memoria con un tempo di risposta di 40 nsec *dal momento in cui gli indirizzi sono disponibili*
- tracciare e illustrare il diagramma di temporizzazione di un di bus asincrono con linee separate per dati e indirizzi e segnali di $\overline{\text{MREQ}}$, $\overline{\text{RD}}$, $\overline{\text{MSYN}}$ e $\overline{\text{SSYN}}$, per una lettura da una memoria con un tempo di risposta di 50 nsec *dal momento in cui gli indirizzi sono disponibili*

Si assuma in entrambi i casi di lavorare in condizioni ideali (nessun ritardo nell'asserimento dei segnali)

Soluzione esercizio su BUS sincrono



Soluzione esercizio su BUS asincrono



Esercizio su BUS

Con riferimento al funzionamento dei bus di un calcolatore:

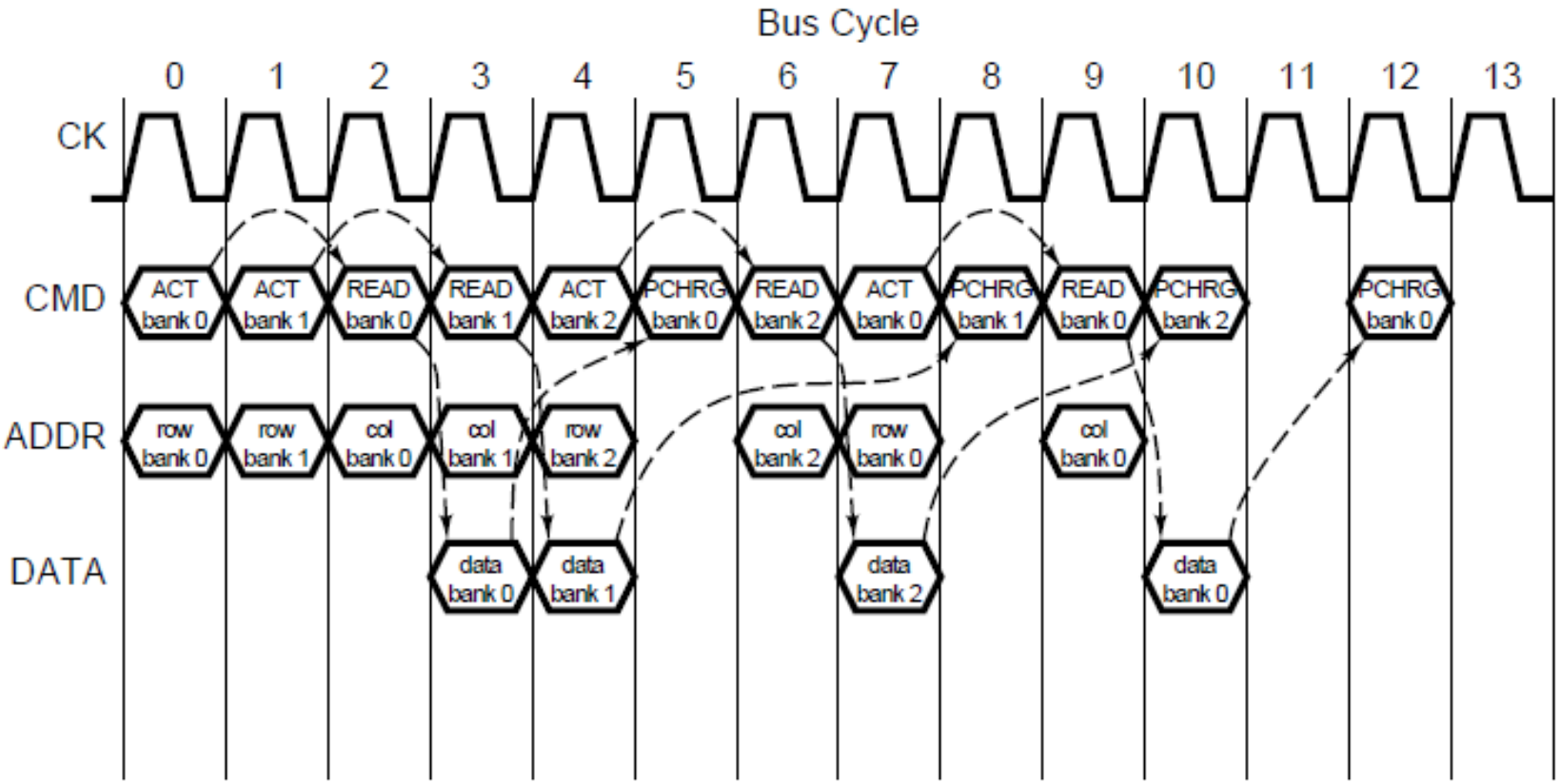
- tracciare e illustrare il diagramma di temporizzazione di un bus sincrono con linee separate per dati e indirizzi che lavora alla frequenza di 50 Mhz, per una lettura da una dispositivo I/O con un tempo di risposta di 100 nsec;
- tracciare e illustrare il diagramma di temporizzazione di un bus asincrono con linee condivise per dati e indirizzi per una scrittura in una memoria con un tempo di risposta di 30 nsec.

Esercizio su BUS

Si consideri un bus di memoria per una CPU a 32 bit che lavora a 500Mhz e funziona in pipeline.

- individuare una ragionevole decomposizione di una transazione (trasferimento di dati tra CPU e memoria) facendo riferimento alle relative linee del BUS;
- indicare mediante un diagramma la temporizzazione dei segnali sul BUS secondo le scelte fatte al punto precedente;
- dire in che cosa consiste il problema del bus skew e come è possibile ovviare a questo problema.

Spunto per possibile soluzione



Esercizio sull'arbitraggio di bus

Riferendosi agli schemi di arbitraggio dei bus discussi a lezione, indicare le affermazioni esatte tra le seguenti.

- Nell'arbitraggio decentralizzato non è necessaria la linea che segnala l'occupazione del bus. FALSO
- Nell'arbitraggio centralizzato è possibile cambiare i livelli di priorità via software. FALSO
- Nell'arbitraggio centralizzato non è possibile che due dispositivi si prenotino contemporaneamente asserendo la linea di request. FALSO
- Nell'arbitraggio decentralizzato ai fini delle attese la posizione fisica dei dispositivi è comunque ininfluente. FALSO
- Nell'arbitraggio centralizzato a più livelli di priorità un dispositivo può dover attendere un tempo indefinitamente lungo. VERO
- Per motivi di imparzialità l'arbitro è sempre un dispositivo diverso ed esterno al microprocessore. FALSO
- La priorità dei dispositivi di I/O è generalmente più alta della CPU. VERO

Esercizio sull’arbitraggio di bus

Riferendosi agli schemi di arbitraggio dei bus discussi a lezione, indicare le affermazioni esatte tra le seguenti.

@NO Nell'arbitraggio decentralizzato non è necessaria la linea che segnala l'occupazione del bus.

@NO Nell'arbitraggio centralizzato è possibile cambiare i livelli di priorità via software.

@NO Nell'arbitraggio centralizzato non è possibile che due dispositivi si prenotino contemporaneamente asserendo la linea di request.

@NO Nell'arbitraggio decentralizzato ai fini delle attese la posizione fisica dei dispositivi è comunque ininfluente.

@SI Nell'arbitraggio centralizzato a più livelli di priorità un dispositivo può dover attendere un tempo indefinitamente lungo.

@NO Per motivi di imparzialità l'arbitro è sempre un dispositivo diverso ed esterno al microprocessore.

@SI La priorità dei dispositivi di I/O è generalmente più alta della CPU.

Esercizio sui protocolli di bus

Considerando i protocolli di bus discussi a lezione, indicare le affermazioni esatte tra le seguenti.

- Nel protocollo sincrono il trasferimento dei dati tra ogni coppia master-slave avviene sempre al terzo ciclo di clock. FALSO
- In una operazione di lettura da memoria in un bus sincrono, il segnale di memory request è asserito dal master. VERO
- Nel protocollo sincrono non è possibile che le linee dati e indirizzi siano condivise. FALSO
- I block transfer aumentano la banda del bus perchè consentono di trasmettere dati tra lo stesso master e più slave. FALSO
- Nel protocollo asincrono il segnale di master synchronization viene negato in risposta a slave synchronization, dopo la lettura dei dati. VERO
- Nel protocollo asincrono il segnale di slave synchronization viene negato in risposta all'asserzione di master synchronization. FALSO
- Un segnale asserito è negato quando è a V_{cc}. VERO

Esercizio sui protocolli di bus

Considerando i protocolli di bus discussi a lezione, indicare le affermazioni esatte tra le seguenti.

@NO Nel protocollo sincrono, il trasferimento dei dati tra ogni coppia master-slave avviene sempre in un prestabilito ciclo di clock.

@SI In una operazione di lettura da memoria in un bus sincrono, il segnale di memory request è asserito dal master.

@NO Nel protocollo sincrono non è possibile che le linee dati e indirizzi siano condivise.

@NO I block transfer aumentano la banda del bus perchè consentono di trasmettere dati tra lo stesso master e più slave.

@SI Nel protocollo asincrono il segnale di master synchronization viene negato in risposta a slave synchronization, dopo la lettura dei dati.

@NO Nel protocollo asincrono il segale di slave synchronization viene negato in risposta all'asserzione di master synchronization.

@SI Un segnale asserito basso è negato quando è a V_{cc}.

Esercizio sull'arbitraggio di bus

Con riferimento agli schemi di arbitraggio dei bus, indicare se le seguenti affermazioni sono vere o false.

- Nell'arbitraggio decentralizzato non è possibile introdurre un meccanismo di priorità. VERO
- Nell'arbitraggio centralizzato l'arbitro conosce di volta in volta quanti dispositivi richiedono l'uso del bus. FALSO
- Un arbitraggio centralizzato con una sola linea di richiesta non può gestire priorità. VERO
- Nell'arbitraggio decentralizzato la priorità non dipende dalla posizione fisica dei dispositivi. FALSO
- Nell'arbitraggio centralizzato non è necessaria la linea busy che segnala che il bus è impegnato. VERO
- Un arbitraggio decentralizzato non richiede un dispositivo dedicato di arbitraggio. VERO
- Nell'arbitraggio centralizzato può succedere che un dispositivo che non ha richiesto il bus riceva il segnale di grant. VERO
- La priorità della CPU nell'uso del bus è sempre più alta di quella dei dispositivi di I/O. FALSO

Esercizio sull’arbitraggio di bus

Con riferimento agli schemi di arbitraggio dei bus, indicare se le seguenti affermazioni sono vere o false.

@SI Nell'arbitraggio decentralizzato non è possibile introdurre un meccanismo di priorità.

@NO Nell'arbitraggio centralizzato l'arbitro conosce di volta in volta quanti dispositivi richiedono l'uso del bus.

@SI Un arbitraggio centralizzato con una sola linea di richiesta non può gestire priorità.

@NO Nell'arbitraggio decentralizzato la priorità non dipende dalla posizione fisica dei dispositivi.

@SI Nell'arbitraggio centralizzato non è necessaria la linea busy che segnala che il bus è impegnato.

@SI Un arbitraggio decentralizzato non richiede un dispositivo dedicato di arbitraggio.

@SI Nell'arbitraggio centralizzato può succedere che un dispositivo che non ha richiesto il bus riceva il segnale di grant.

@NO La priorità della CPU nell'uso del bus è sempre più alta di quella dei dispositivi di I/O.

Esercizio

Si vuole realizzare un piccolo calcolatore embedded a 16 bit dotato di una ROM di 4KB, una RAM di 8KB e un dispositivo di I/O memory mapped a cui interfacciarsi tramite una PIO

- Definire gli spazi di indirizzamento dei vari dispositivi a disposizione supponendo di poter utilizzare 16 bit per specificare gli indirizzi;
- Indicare come è possibile ottenere con porte logiche i corretti segnali di chip select a partire dalle linee del bus degli indirizzi;
- Indicare come è possibile semplificare i circuiti determinati al punto B con una decodifica parziale degli indirizzi.

Una soluzione

ROM: 4K (2^{12}) indirizzi

- Scelta:
  - da 0000 0000 0000 0000
  - a 0000 1111 1111 1111
- Distinguibile da i primi 4 bit a 0000

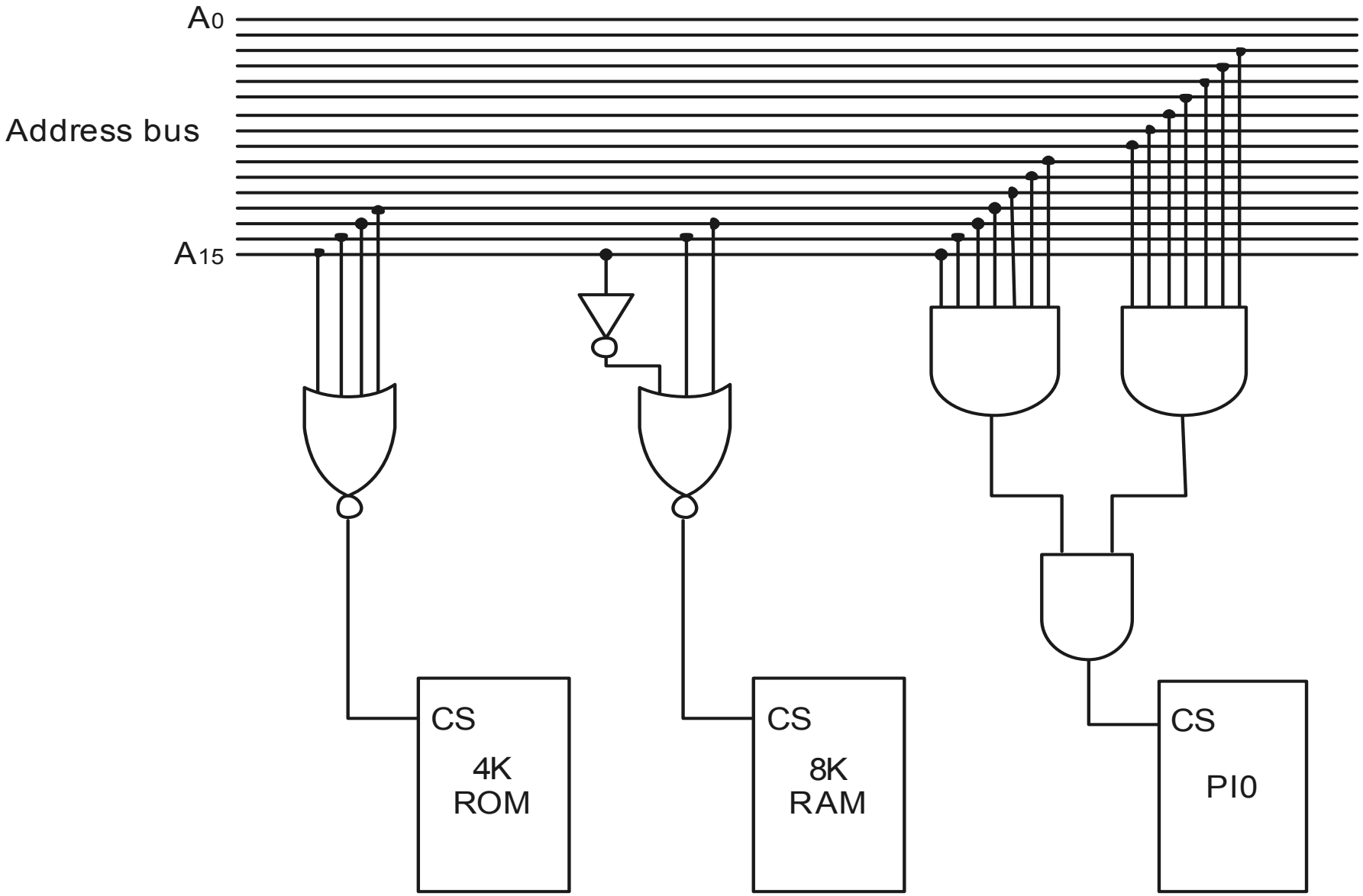
RAM: 8K (2^{13}) indirizzi

- Scelta:
  - da 1000 0000 0000 0000
  - a 1001 1111 1111 1111
- Distinguibile da i primi 3 bit a 100

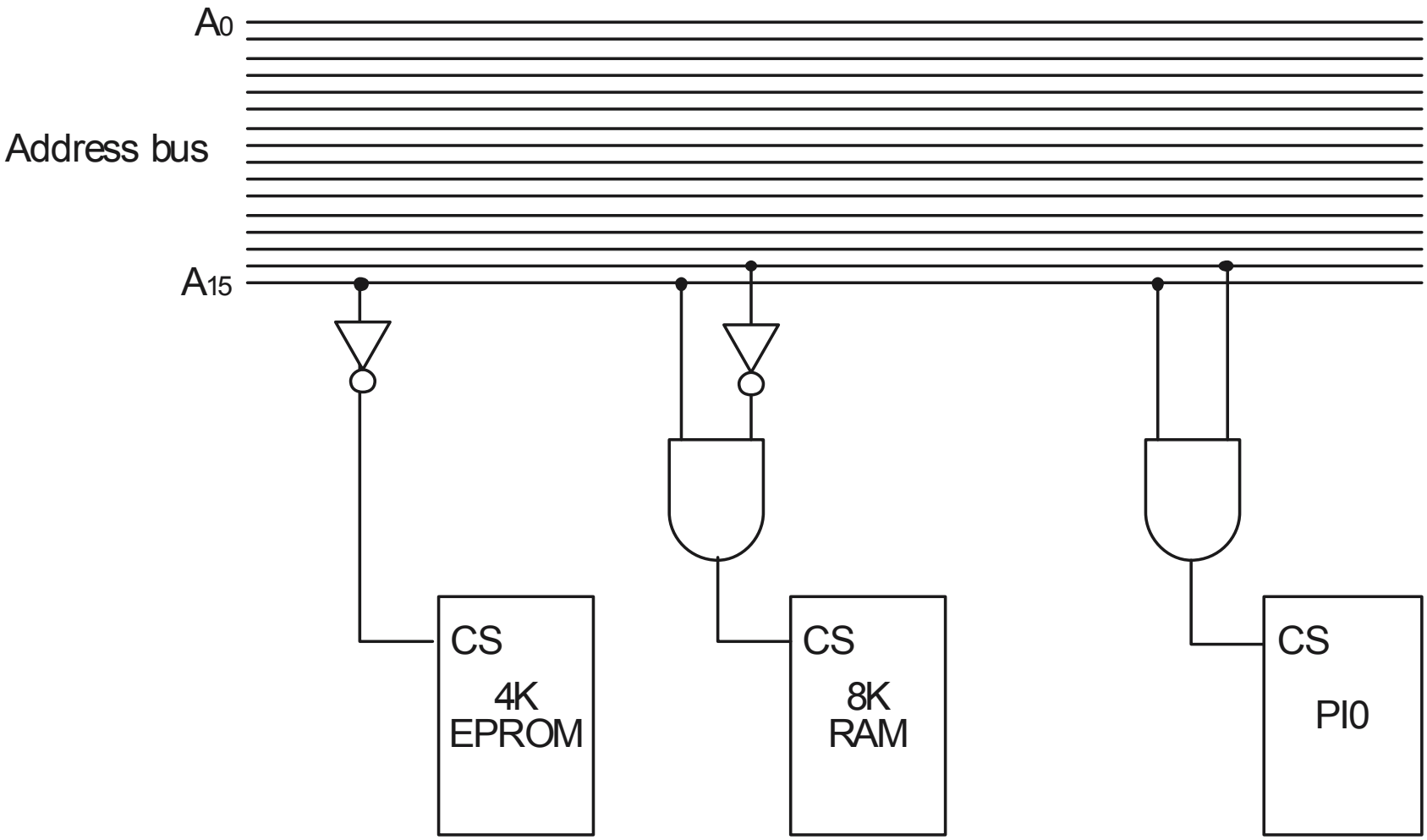
PIO: 4 indirizzi

- Scelta:
  - da 1111 1111 1111 1100
  - a 1111 1111 1111 1111
- Distinguibile da i primi 14 bit tutti a 1

Decodifica completa



Decodifica parziale



Esercizio

Si vuole realizzare un piccolo calcolatore embedded a 16 bit dotato di una ROM di 8KB, una RAM di 16KB e un dispositivo di I/O memory mapped a cui interfacciarsi tramite una PIO (dotato di 4 porte indirizzabili).

- definire gli spazi di indirizzamento dei vari dispositivi a disposizione supponendo di poter utilizzare tutti i 16 bit per specificare gli indirizzi;
- indicare come è possibile ottenere con porte logiche i corretti segnali di chip select a partire dalle linee del bus degli indirizzi;
- indicare come è possibile semplificare i circuiti determinati al punto B con una decodifica parziale degli indirizzi.

Esercizio

Un microprocessore embedded a 8 bit deve comunicare mediante un bus parallelo con una ROM di 64B e, facendo uso di due schede PIO, con 6 dispositivi di I/O memory mapped (3 per scheda).

- definire gli spazi di indirizzamento dei vari dispositivi a disposizione supponendo di poter utilizzare tutti gli 8 bit per specificare gli indirizzi;
- indicare come è possibile ottenere con porte logiche i corretti segnali di chip select a partire dalle linee del bus degli indirizzi;
- indicare come è possibile semplificare i circuiti determinati al punto B con una decodifica parziale degli indirizzi.