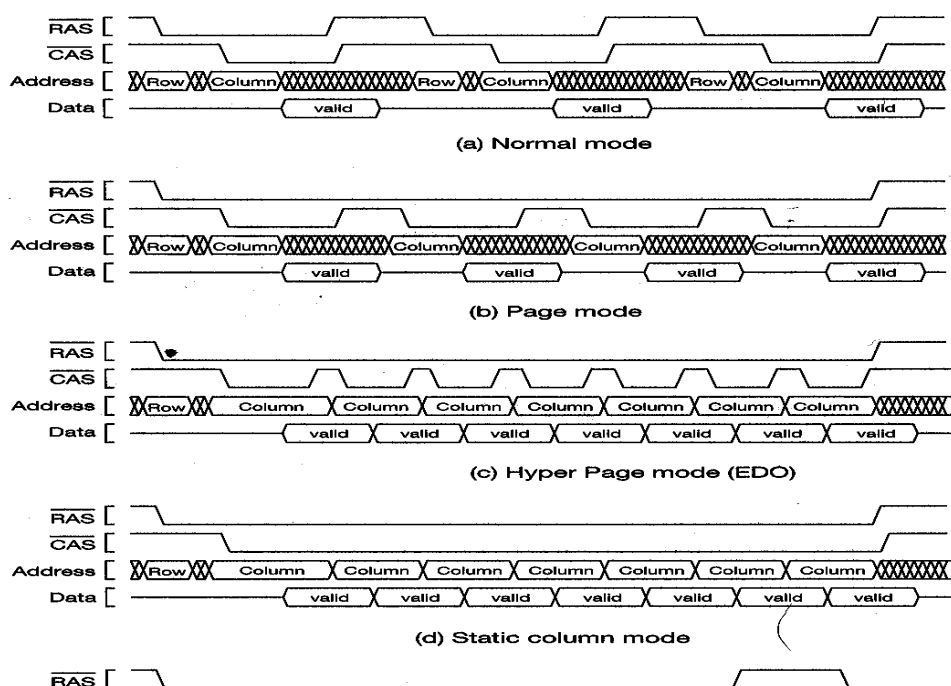


## FAST OPERATIVE

Normalmente le letture/scritture effettuate su una memoria sono dovuti all'aggiornamento delle cache. Questo significa che gli indirizzi sono tra loro contigui.

Inoltre l'architettura interna di una DRAM (struttura a riga e a colonne dove a ogni incrocio è presente un condensatore e un pass transistor che ne permette la lettura) fa sì che per attivare una cella bisogna attivare tutta la riga e tutta la colonna. In questo modo solo una cella sarà selezionata sia su riga che su colonna. Le selezioni sono fatte da due multiplexer, uno per le righe, l'altro per le colonne.

Tenendo fisso l'ingresso su un multiplexer (che seleziona dunque sempre la stessa riga o colonna) e variando l'altro si ha la possibilità di leggere indirizzi contigui di memoria. Questo è il principio cardine del Fast Operative.



Si hanno diverse tipologie di Fast Operative e queste possono essere sia sincrone che asincrone. Un Fast Operative mode asincrono non è sincronizzato con il clock del bus.

Nel Fast Operative Page Mode si mantiene fisso il RAS e si legge un'intera riga di memoria (agendo sul CAS). E' ovvio il risparmio di tempo rispetto a meccanismi di lettura tradizionali (anche perché posso non aspettare il tempo di precarica tra la lettura dei vari bit).

E' ovvio che per la gestione di questi meccanismi si ha la necessità di controller decisamente più sofisticati.

Le SD RAM e le DDR RAM sono memorie che supportano il FAST OPERATIVE MODE sincrono. In queste tecnologie si fa in modo che l'operatività della RAM dinamica sia coerente e sincronizzata con la tempistica del bus. Essendo sincrone è inoltre possibile sapere a priori quanto tempo ci impiegherà una data operazione sulla memoria. Proprio perché questo tempo è certo è possibile innescare un meccanismo di parallelismo: si attiva una lettura o scrittura sulla memoria e durante il suo svolgimento (dove si hanno un certo numero di operazioni interne ai banchi) si possono eseguire altri compiti (il bus risulta infatti libero) avendo cura di recuperare il risultato dell'operazione sulla memoria quando pronto.

Nelle memorie sincrone ci sono una serie di configurazioni programmabili in un registro apposito all'interno del chip. Il registro di controllo determina le modalità operative può essere programmato

attraverso apposite configurazioni dei pin.

## DDR

L'evoluzione delle DDR riguarda principalmente la tempistica. DDR significa infatti Double Data Rate cioè per ogni ciclo di clock si leggono due dati.

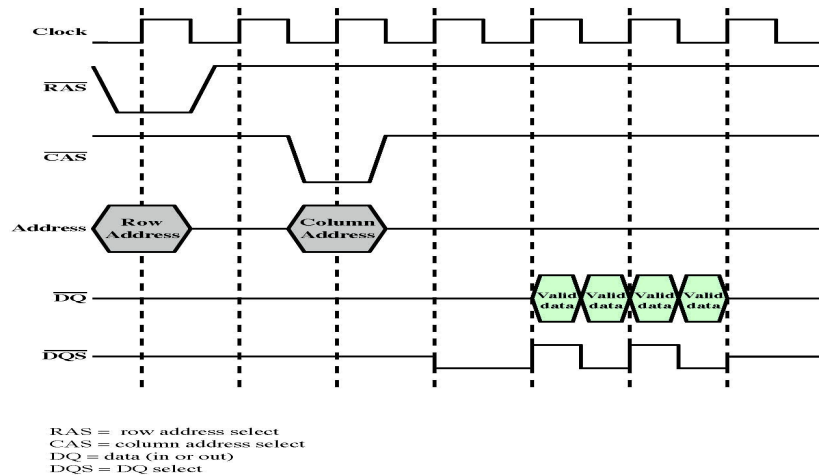
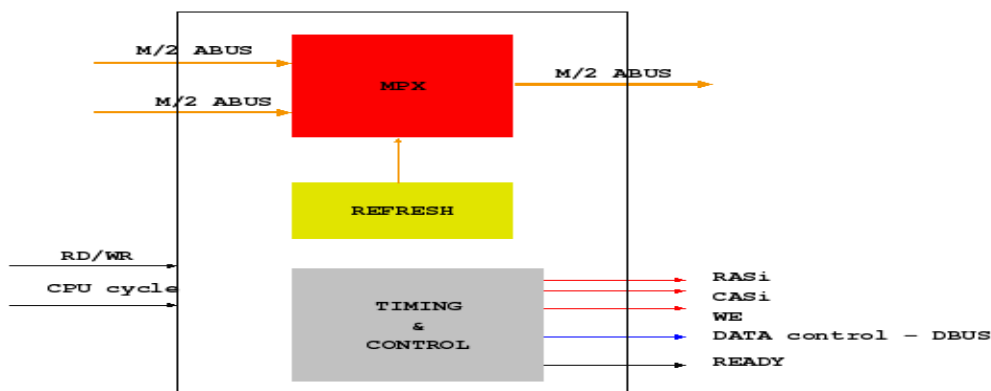


Figure 5.15 DDR SDRAM Read Timing

## DRAM CONTROLLER

Un sottosistema di memoria è basato essenzialmente su un DRAM controller presente nel chipset oppure su un chip dedicato.

- Un DRAM Controller contiene essenzialmente tre tipi di blocchi:
- Un multiplexer per il multiplexaggio dell'address bus.
- Un blocco per il refresh: un contatore che gestisce la temporizzazione del refresh
- Timing and Control: riceve dal bus informazioni come il tipo di ciclo, temporizzazione, lettura o scrittura,... e le processa generando i segnali interni necessari (ras, Cas,...). È il vero cuore del DRAM Controller.



## ORGANIZZAZIONE FISICA DDR

Fisicamente, una DDR è fisicamente organizzata in:

- Banco: blocco fisico/logico con parallelismo di un byte selezionato dai segnali BEi\* (*Bank Enable*). Il numero dei banchi dipende dal parallelismo del DBUS.
- Device: singolo chip caratterizzato da una profondità (numero di indirizzi) e da un parallelismo dati (bit di dato).
- Modulo: insieme di device con parallelismo pari al DBUS (tutti i banchi) realizzato su unica piastrina (SIMM, DIMM,...).
- Rank: porzione di un modulo costituito da un blocco di device con parallelismo pari al DBUS e profondità pari a quelle dei chip.

## CONTROLLO DI ERRORE

Con il crescere della dimensione delle memorie la probabilità che si verifichi un errore costante o temporaneo cresce. E' necessario disporre di meccanismi che rilevino (codici di parità) ed eventualmente correggano (ECC) queste situazioni.

La rilevazione dell'errore di parità sul byte necessita di memorizzare un bit aggiuntivo per ogni byte memorizzato. Per far ciò bisogna disporre di circuiti che nella fase di scrittura calcolino questo bit e lo scrivano, una cella di memoria in più per ogni byte (si memorizzano infatti 9 bit per ogni byte), circuiti di lettura che ricalcolino la parità ed effettuino il controllo con il codice memorizzato, meccanismo a livello di sistema in caso di errore rilevato.

N.B. il codice di parità non corregge gli errori e rileva solo un numero di errori dispari, non pari.

Attraverso i codici di Hamming è possibile rilevare e correggere gli errori. Nei moderni calcolatori si utilizzano codici single error correction double error detection.

E' ovvio che la politica di controllo dell'errore diminuisce le prestazioni delle memorie.

L'informazione di presenza di errore viene inviata a un interrupt prioritario o in altre architetture a un interrupt non mascherabile. Cosa succeda una volta attivato questo interrupt dipende dal sistema operativo. Si hanno due casi:

- Si informa l'utente dell'errore con un messaggio
- Si riconfigura il sistema abortendo i processi che fanno riferimento al modulo in cui è presente l'errore e disabilitando quel modulo (soluzione più raffinata della precedente)

Negli ultimi processori per facilitare l'organizzazione della memoria è presente la generazione automatica dei bit di parità. Oltre a emettere i dati sono anche emessi un certo numero di bit di parità (uno per byte). I bit di parità possono anche essere acquisiti se l'operazione in questione è la lettura.

Nelle memorie più avanzate è presente la correzione con l'ECC. Si hanno dei banchi di memoria aggiuntivi per la sua memorizzazione.

# SOTTOSISTEMA DI I/O NELL'ARCHITETTURA 80X86

Un sistema di gestione di un periferico è composto da 3 livelli.

- Il primo livello è il bus di sistema (realizza le transazioni conseguenti alle istruzioni di input/output). Lo spazio di I/O determina il numero di indirizzi attribuiti all'insieme dei registri associati ai periferici.  
Ci deve essere coerenza tra dispositivi e bus.
- Il secondo livello riguarda la gestione delle modalità di trasferimento dati che possono essere legate a meccanismi di interruzione o a meccanismi di DMA. Il bus deve ovviamente supportare questi meccanismi (per esempio sull'host bus non possono essere connessi periferici in quanto non sono presenti i segnali per la gestione delle interruzioni).
- Il terzo livello riguarda l'interfaccia, cuore fondamentale per il funzionamento dei periferici. E' in grado di attivare i periferici quando richiesto (ad esempio quando vengono indirizzati dal bus).

All'interno dell'interfaccia sono presenti una serie di registri che permettono la gestione del periferico e del dialogo con il bus. I registri sono di solito 3:

- + registro di comando: specifica la funzione da realizzare
- + registro di stato: riporta la condizione del periferico
- + registro di dato: permette di trasferire i dati dal periferico al sistema e

viceversa

Esistono due principali filosofie per la connessione dei periferici

- memory mapped: i periferici rispondono a particolari indirizzi nello spazio di indirizzamento della memoria
- isolated I/O: esiste un set di istruzioni specifico per il dialogo con il periferico. Lo spazio di indirizzamento è separato da quello della memoria.

Gli indirizzi presenti sul bus indirizzi sono in parte veicolati a un decoder I/O che va a selezionare l'interfaccia opportuna e in parte direttamente all'interfaccia per la selezione interna dei registri.