**Tema 6: Jerarquía de Memorias: Memoria Principal**

**Introducción**

Modelo de Memoria Principal:

* La Memoria Principal (MP) puede verse como:

byte M[MemSize];

* Operaciones básicas:
  + Lectura: dato = M[direccion];
  + Escritura: M[direccion] = dato;
* ¡Atención! Desde el punto de vista del programador:
  + La MP se direcciona a nivel de byte.
  + Los accesos a memoria pueden ser de múltiples tamaños: 1, 2, 4 u 8 bytes.
  + Si leemos 4 bytes en la dirección X, accedemos a las direcciones X, X+1, X+2 y X+3.
  + Little endian vs big endian.

Tipos de Memorias

En función de la **perdurabilidad**:

* Volátil (memoria cuya información se pierde al interrumpirse el flujo eléctrico; DRAM, RAM…).
* No Volátil (memoria que no necesita energía para mantener guardada la información en ella; EEPROM…).

En función del **tipo de acceso**:

* Sólo lectura (ROM, Read Only Memory).
* Lectura / Escritura (RAM, Random Access Memory).

En función del **tipo de uso**:

* Primaria (semiconductores).
* Secundaria (dispositivos de almacenamiento E/S, magnéticos y ópticos).

En función de la **forma de acceso**:

* Memorias de Acceso Secuencial (cinta VHS).
* Memorias de Acceso Directo (DVD).

**Memorias de Semiconductores**

Tipos:

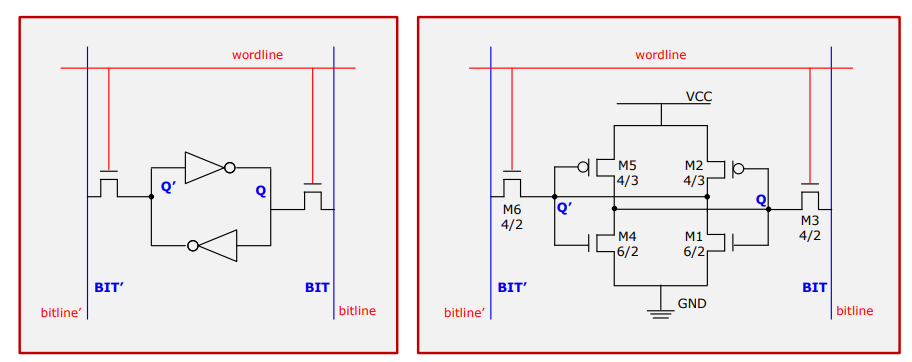
1. Memoria Estática (SRAM, Static RAM): Cada celda de memoria equivale a 1 biestable (6-8 transistores). En comparación con las DRAM:
   1. son rápidas.
   2. tienen un alto consumo.
   3. pequeñas (poca capacidad).
   4. caras.

**-> Memoria Cache**

1. Memoria Dinámica (DRAM, Dynamic RAM): Cada celda se comporta como un condensador (1-1.x transistores). En comparación con las SRAM:
   1. son lentas.
   2. tienen un bajo consumo.
   3. grandes (mucha capacidad).
   4. baratas.
   5. Problema del refresco.

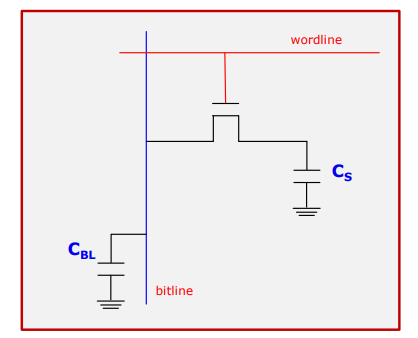
**-> Memoria Principal**

Celda SRAM de 6 Transistores:



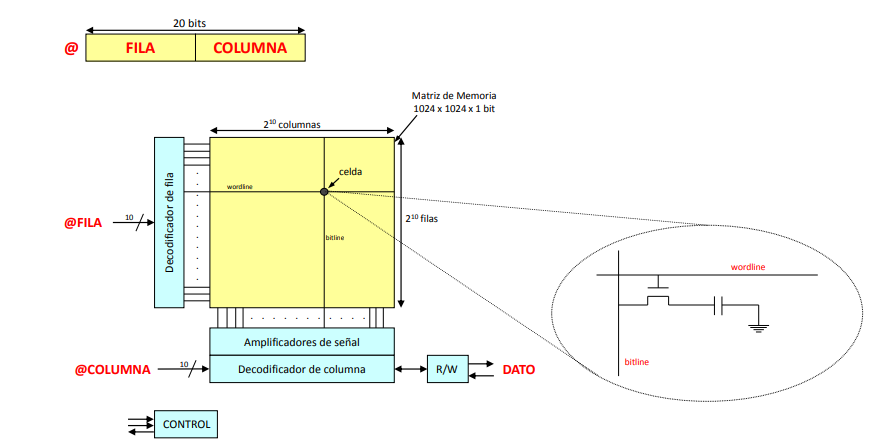
* La información se almacena en 2 inversores acoplados.
* Al activar la wordline el dato almacenado se lee a través de las bitlines.
* Se obtiene el dato negado y sin negar.

Celda DRAM de 1 Transistor:



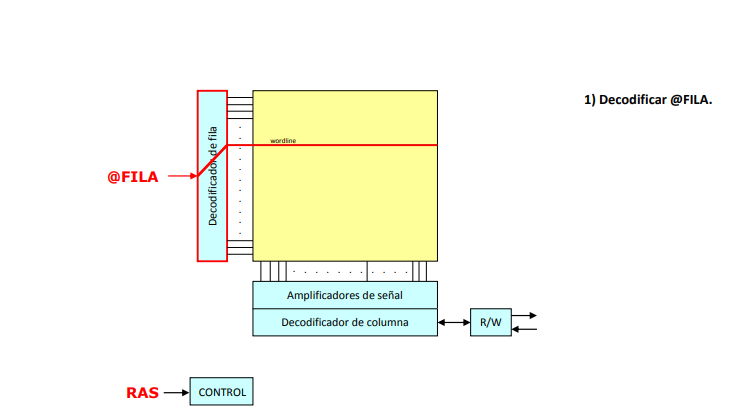
* La información se almacena en el condensador CS.
* Al activar la wordline el dato almacenado en CS se lee a través de la bitline.
* El condensador se va descargando poco a poco, es necesario recargarlo regularmente (refresco).

**Estructura interna de una DRAM**

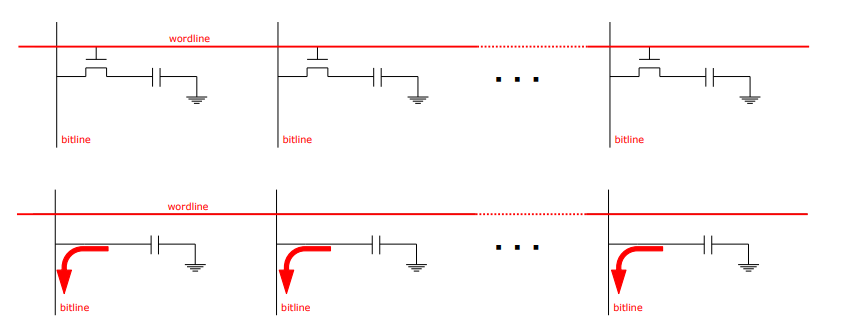


Una operación de lectura:

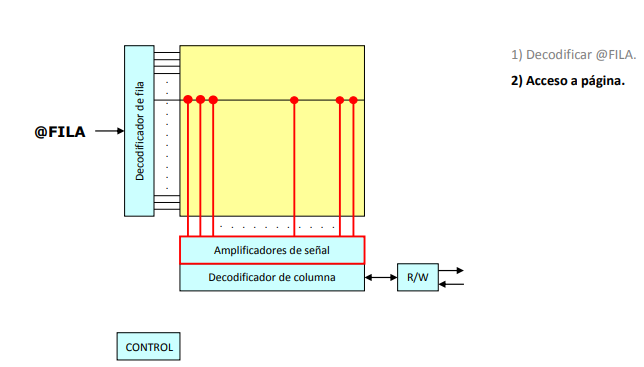
1) Decodificar @FILA, se activa la señal wordline.



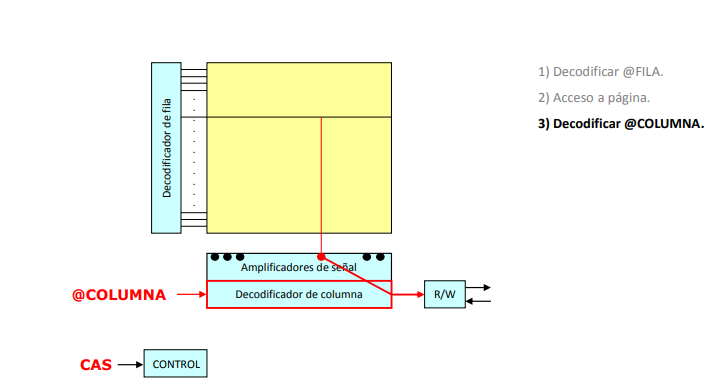
2.1) Se accede a todas las celdas de la fila, los datos de toda la fila se envían a los amplificadores de señal y se recupera la tensión (el dato está en un condensador que se va descargando poco a poco).



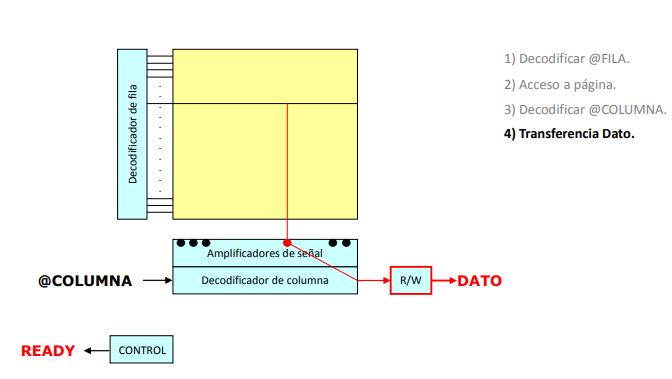
2.2) Se accede a todas las celdas de la fila, los datos de toda la fila se envían a los amplificadores de señal y se recupera la tensión.



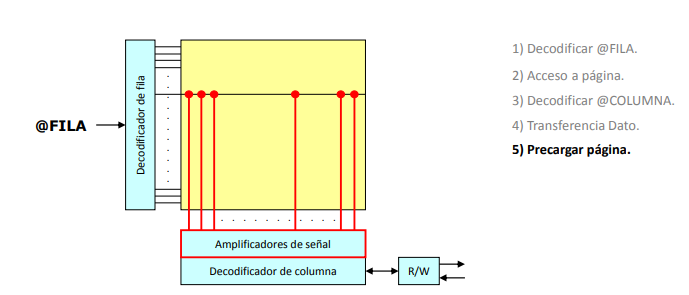
3) Decodificar @COLUMNA, se selecciona una bitline y se envía el dato al buffer R/W.



4) Se envía el dato al exterior desde el buffer R/W.

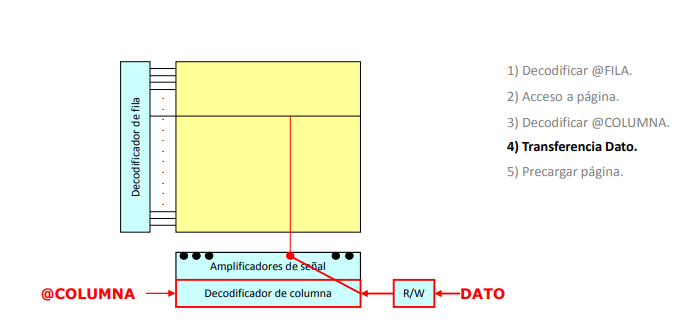


5) La lectura es destructiva, hay que reescribir la celda (y toda la fila) para recuperar el valor original y precargar los bitlines para el siguiente acceso a memoria.

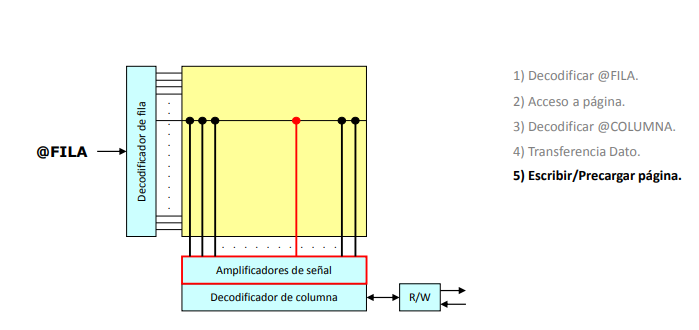


Una operación de escritura:

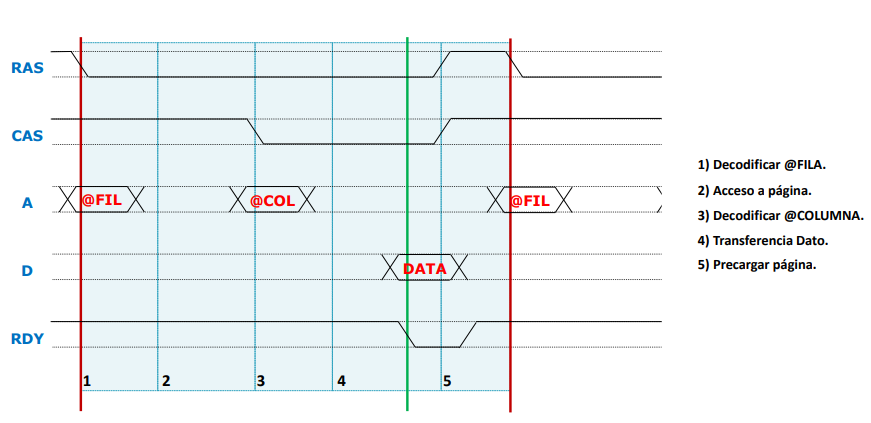
3) y 4) Prácticamente igual, la única diferencia es que la celda se reescribe con el dato que entra por el buffer R/W.



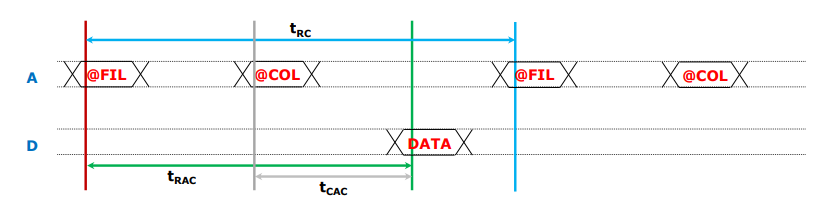
5) Hay que reescribir la celda con el nuevo valor (y el resto de la fila con el valor original).



Cronograma simplificado de una lectura:

****

Valores principales en una lectura a DRAM:



1. Tiempo de acceso (tRAC): retardo máximo desde que se suministra la dirección de fila hasta que se obtiene el dato → latencia de memoria.

2. Tiempo de ciclo (tRC): intervalo de tiempo mínimo entre dos accesos consecutivos a memoria → ancho de banda.

3. Tiempo de acceso a columna (tCAC): retardo máximo desde que se suministra la dirección de columna hasta que se obtiene el dato.

**Evaluación y Optimización**

La memoria principal suele estar organizada en DIMMs.

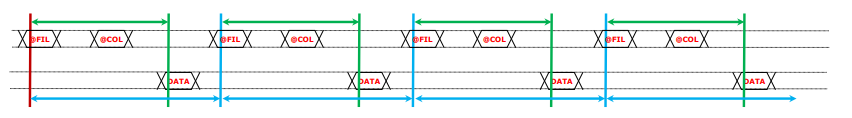
Se accede en paralelo a todos los chips. En un acceso se accede (p.e.) a 64 bits (8B).

¡Todos los accesos a memoria principal son para leer/escribir líneas de cache!

Tamaño típico de una línea de cache: 32B.

Si queremos leer una línea de cache: ¡Es necesario realizar 4 accesos consecutivos a memoria!

Leer una línea de cache: 4 lecturas de memoria principal.

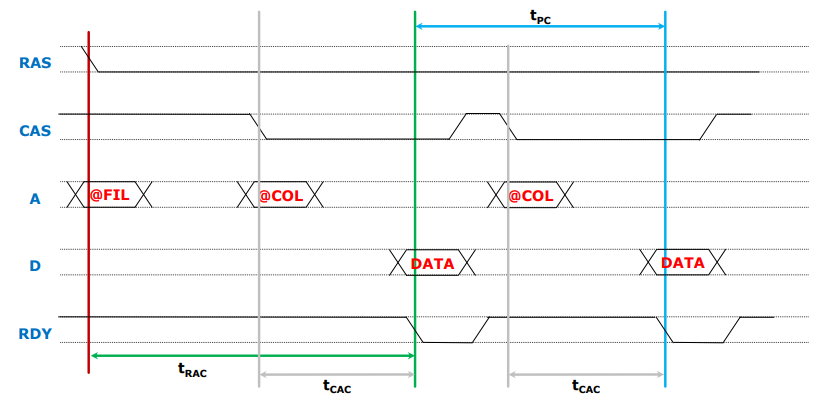
* Tiempo de acceso: 50ns.
* Tiempo de ciclo: 60ns.

Coste de leer una línea de cache:

* Tiempo total: Tiempo de ciclo \* 4 = 240 ns.
* Ancho de banda: 32B / 240ns = 133,33 MB/s.

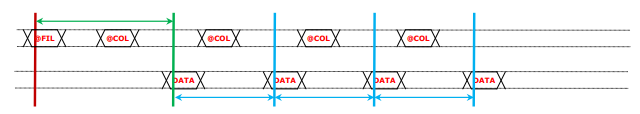
**Posible optimización**: los datos que leemos de memoria componen una línea de cache y están en posiciones consecutivas de memoria → Están en la misma fila del array de memoria (en posiciones consecutivas).

Idea Fundamental: una vez accedida la fila, se puede acceder a varias columnas simplemente cambiando la @COL → Aprovechamos la localidad espacial.



Leer una línea de cache: 4 lecturas de memoria principal.

* Tiempo de acceso: 50ns.
* Latencia de columna: 30ns.



Coste de leer una línea de cache:

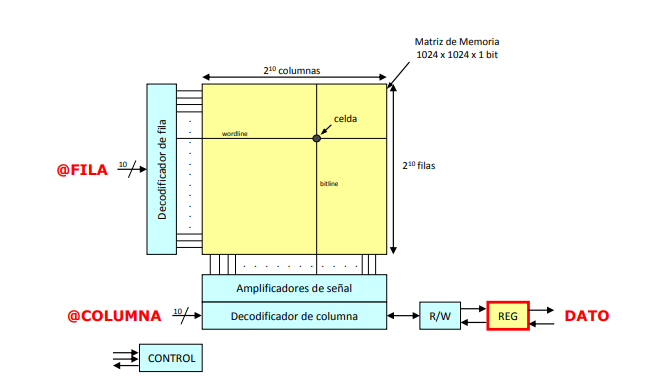
* Tiempo total: Tiempo de acceso + Latencia de columna \* 3 = 140 ns.
* Ancho de banda: 32B / 140ns = 228,57 MB/s.
* Ancho de banda de pico: 8B / 30ns = 266,67 MB/s.

La latencia del primer dato se mantiene.

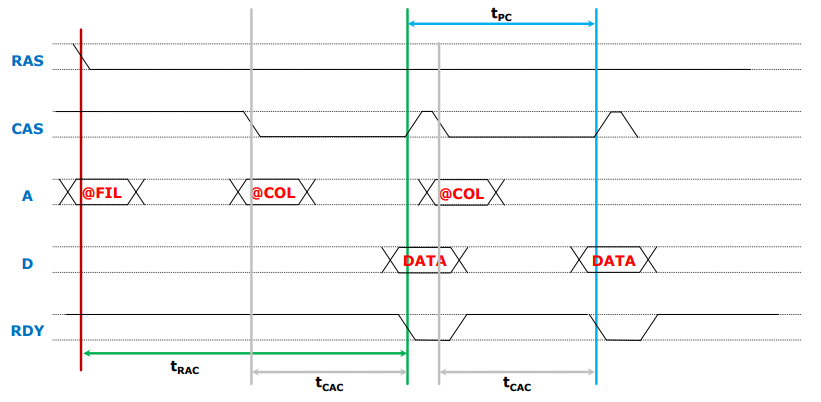
Mejora la latencia de los siguientes datos simplemente cambiando el protocolo de acceso a memoria. El hardware es exactamente el mismo.

Problema: hay que esperar a que el dato sea leído antes de enviar la nueva @COL.

Solución: se añade un registro en la salida de datos → se puede solapar el acceso a los datos con el envío de la nueva @COL.

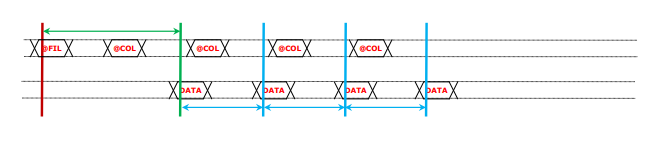


Idea Fundamental: Como el dato está almacenado en un registro, antes de acabar el envío del dato ya podemos enviar la @COL.



Leer una línea de cache: 4 lecturas de memoria principal.

* Tiempo de acceso: 50ns.
* Latencia de columna: 20ns.



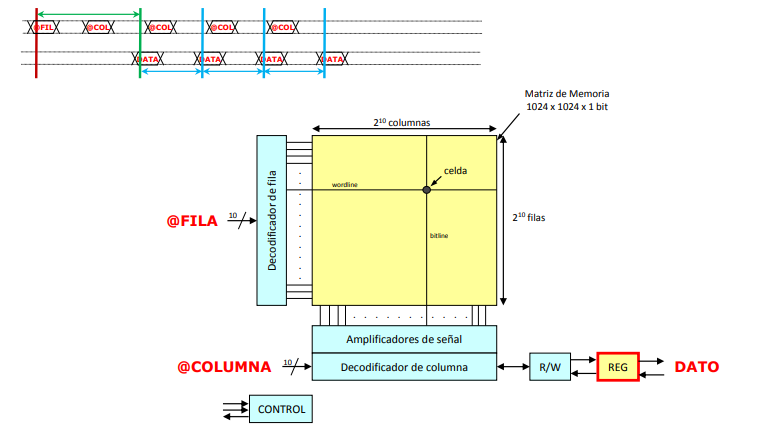
Coste de leer una línea de cache:

* Tiempo total: Tiempo de acceso + Latencia de columna \* 3 = 110 ns.
* Ancho de banda: 32B / 110ns = 290,91 MB/s.
* Ancho de banda de pico: 8B / 20ns = 400 MB/s.

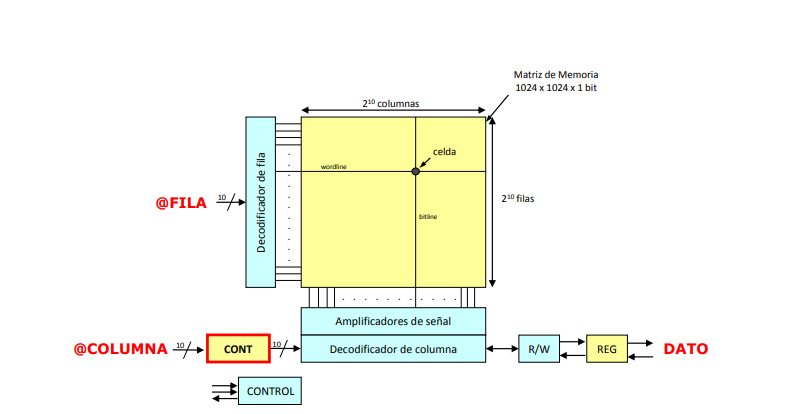
La latencia del primer dato se mantiene.

Mejora la latencia de los siguientes datos simplemente añadiendo un registro.

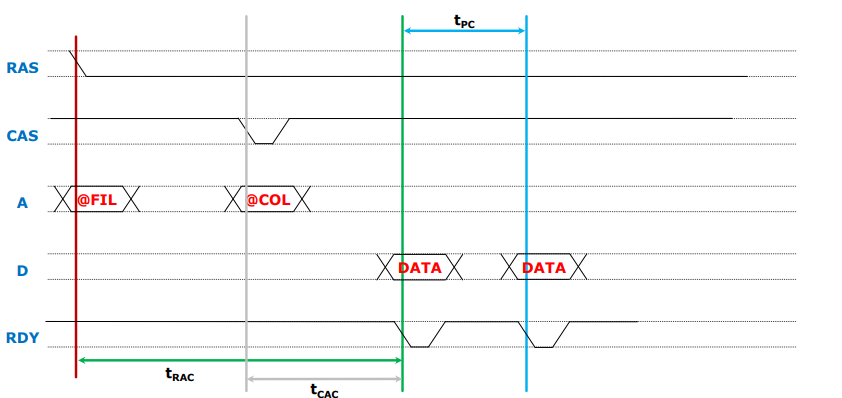
Observación: Hay que enviar la @COL para cada dato. Pero para 1 línea de cache estamos accediendo a @COL, @COL+1, @COL+2 y @COL+3.



**Mejora**: Añadir un contador para que genere de forma automática @COL+1, @COL+2 y @COL+3.

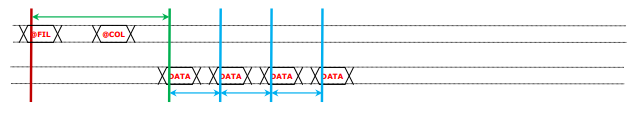


Idea Fundamental: Sólo hay que enviar la @COL una vez, del resto se encarga la propia memoria.



Leer una línea de cache: 4 lecturas de memoria principal.

* Tiempo de acceso: 50ns.
* Latencia de columna: 15ns.



Coste de leer una línea de cache:

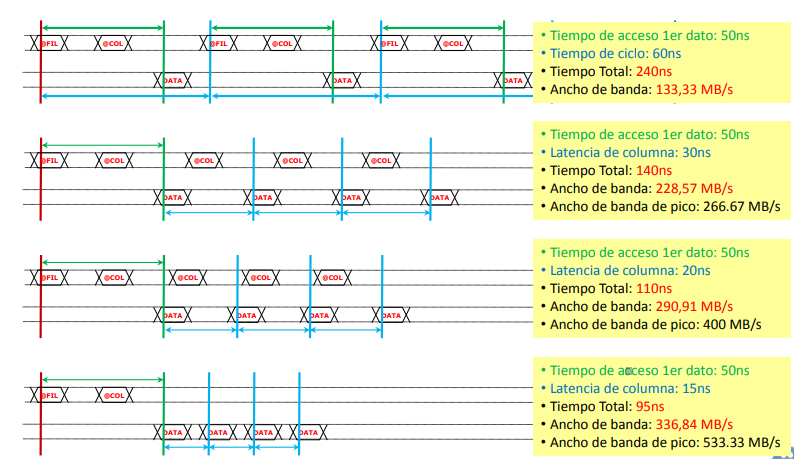
* Tiempo total: Tiempo de acceso + Latencia de columna \* 3 = 95 ns.
* Ancho de banda: 32B / 95ns = 336,84 MB/s.
* Ancho de banda de pico: 8B / 15ns = 533,33 MB/s.

La latencia del primer dato se mantiene.

Mejora la latencia de los siguientes datos simplemente añadiendo un contador.

En este punto se llega al límite de rendimiento de una memoria ASÍNCRONA.

Resumen de las optimizaciones:



Problema de las memorias asíncronas:

El protocolo de comunicación con la memoria es costoso (lento).

Cada paso (de sincronización) requiere:

* Enviar el parámetro (p.e. @FIL).
* Esperar que se estabilice.
* Enviar la señal de sincronismo (p.e. RAS).
* Obtener el parámetro.

Hay 4 pasos de sincronización (FIL, COL, DATA, PRE).

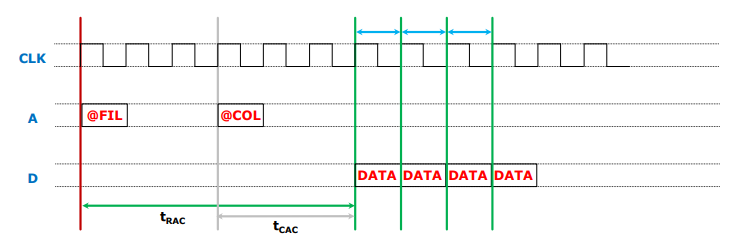
Los tiempos (de acceso, ciclo, etc.) son muy difíciles de reducir por problemas de ruido.

Las memorias asíncronas más veloces se usaban en placas base a 66 MHz.

Solución: Memoria SÍNCRONA:

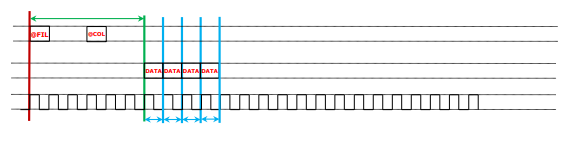
El protocolo se simplifica, se reducen los problemas de ruido.

Se puede aumentar la frecuencia de funcionamiento.



Leer una línea de cache: 4 lecturas de memoria principal.

* Tiempo de acceso: 48ns (6 ciclos).
* Tiempo de transferencia: 8ns (1 ciclo).



Coste de leer una línea de cache:

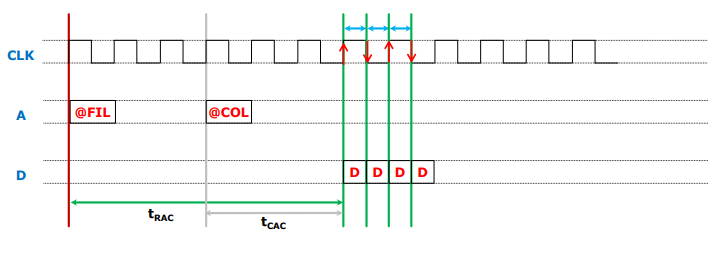
* Tiempo total: Tiempo de acceso + Tiempo de transferencia \* 4 = 80ns.
* Ancho de banda: 32B / 80ns = 400 MB/s.
* Ancho de banda de pico: 8B / 8ns = 1 GB/s.

La latencia del primer dato mejora un poco.

La latencia de los siguientes datos mejora de forma sustancial. Además, el margen de mejora es muy grande.

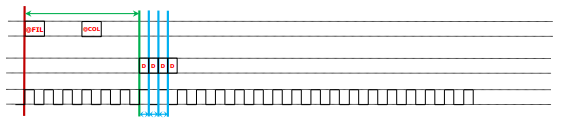
DDR SDRAM:

Mejora: modificando la salida de datos se puede aumentar de forma sustancial el ancho de banda obtenido.

DDR SDRAM: Double Data Rate Synchronous Dynamic Random-Access Memory.

Leer una línea de cache: 4 lecturas de memoria principal.

* Tiempo de acceso: 48ns (6 ciclos).
* Tiempo de transferencia: 4ns (1/2 ciclo).

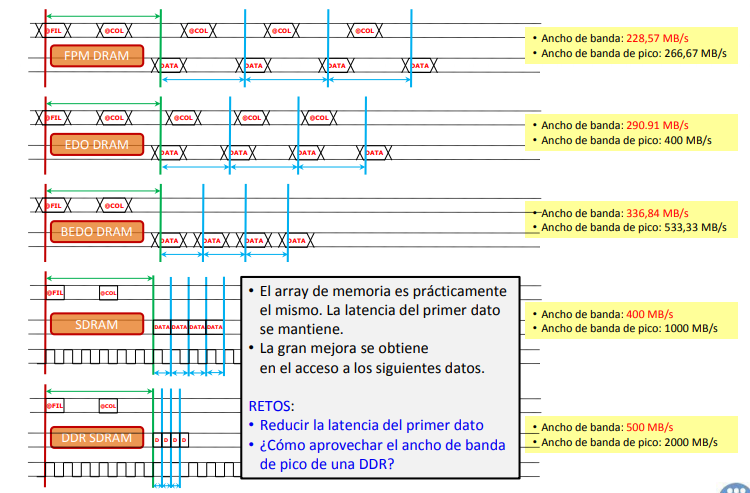


Coste de leer una línea de cache:

* Tiempo total: Tiempo de acceso + Tiempo de transferencia \* 4 = 64ns.
* Ancho de banda: 32B / 64 ns = 500 MB/s.
* Ancho de banda de pico: 8B / 4ns = 2 GB/s.

La latencia del primer dato es la misma.

Un “pequeño” cambio en el hardware de la salida de datos permite doblar el ancho de banda de pico de la memoria.

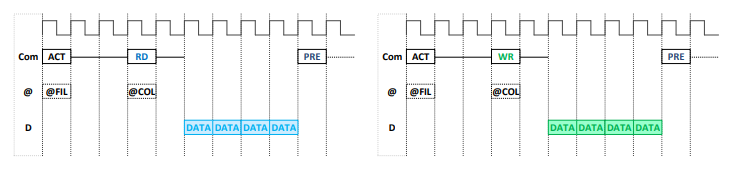


Funcionamiento estándar de una SDRAM:

Todas las SDRAM, incluyendo los diferentes tipos de DDR, funcionan con comandos:

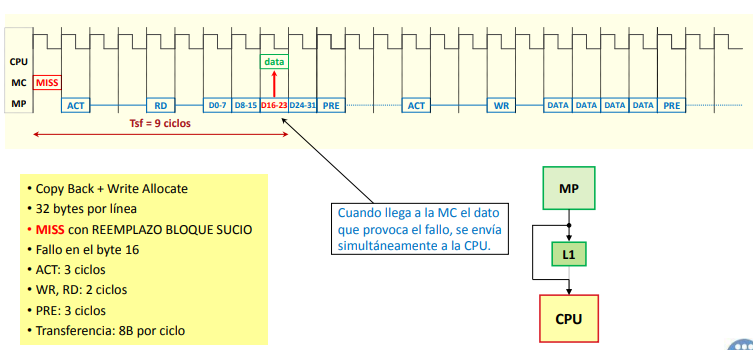
* ACTIVE (ACT).
* READ (RD).
* WRITE (WR).
* PRECHARGE (PRE).

Cronogramas de acceso en Lectura y en Escritura:



Revisitando Cronogramas:

Continuación Anticipada (Early Restart): en cuanto llega el dato que provoca el fallo, se envía al procesador:



Transferencia en desorden: se envía en primer lugar la palabra que ha provocado el fallo:

